(11)

REPORT OF THE DOD JOINT SERVICE TASK FORCE ON SOFTWARE PROBLEMS

AD A123449

DTIC ELECTE JAN 13 1983 S D A

DTIC FILE COPY

This document has been approved for public release and sale; its distribution is unlimited.

Department of Defense
July 30, 1982

83 01 13 014

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A123 449 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Report of the DoD Task Force on Software Problems	5. TYPE OF REPORT & PERIOD COVERED April 23-July 30,1982 / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) The DoD Joint Service Task Force on Software Problems	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy Under Secretary of Defense Research & Advanced Technology Washington, D.C.	12. REPORT DATE July 30, 1982 / 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unrestricted

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Unrestricted

This document has been approved for public release and sale; its distribution is unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Acquisition, change, embedded computer systems, expertise, initiatives, life cycle, management, military supremacy, software, software problems, weapon systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The opportunities and problems posed by computer software embedded in DoD weapon systems were investigated by a joint Service task force. Existing studies were combined with the observations of DoD project managers by the software experienced task force members.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

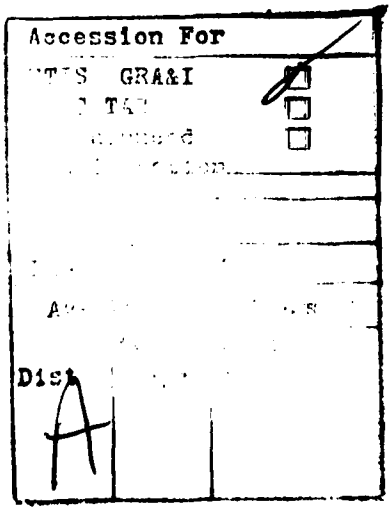
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

The task force concluded that software represents an important opportunity in regard to the military mission. Further, it was concluded that technological excellence in software is an important factor in maintaining U.S. military superiority, but that the many problems facing DoD in software endangers this superiority.

Therefore, it is recommended that the DoD take a leadership role in establishing an embedded computer software initiative with strong joint Service cooperation and which includes both academia and industry as willing participants. The initiatives should address acquisition and management practices, technology R&D and utilization, and development and use of expertise.

RE: Distribution Unrestricted
Document is unlimited per Ms. June Ludwig,
DUSDR&AT/Ada Joint Program Office



DTIC COPY INSPECTED 2

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

REPORT OF THE DOD JOINT SERVICE TASK FORCE ON SOFTWARE PROBLEMS

Department of Defense

July 30, 1982

The DoD Joint Service Task Force on Software Problems was formed at the direction of the Deputy Under Secretary of Defense for Research and Advanced Technology with support from the Assistant Secretary of the Army (RD&A), the Assistant Secretary of the Navy (RE&S), and the Assistant Secretary of the Air Force (RD&L) to identify the problems and opportunities posed by the use of software in computers embedded in DoD weapon systems. This document is the final report of the task force.

The DoD representatives on the task force were:

- Lt. Col. Larry E. Druffel, USAF (Chairman)
  Director, Ada Joint Program Office
  Suite 1210
  801 N. Randolph Street
  Arlington, Virginia 22203

- Mr. Joseph E. Kernan (Army Representative)
  Chief, Software Technology Development Division
  CENTACS, CECOM, Fort Monmouth, New Jersey

- LCDR Kathleen K. Paige, USN (Navy Representative)
  Deputy, AN/UYK-43 Acquisition Project
  Naval Sea Systems Command (PMS 408), Washington, D. C.

- Capt. William A. Riski, USAF (Air Force Representative)
  Digital Systems Engineer
  HQ AFLC/LOEC, Wright-Patterson AFB, Ohio

Support contractors to the task force were:

- Mr. Bufford D. Farmer (Air Force support contractor)
  Engineering Office Site Manager
  TRW Defense Systems Group
  Warner-Robins, Georgia

- Mr. Andrew B. Ferrentino (Navy support contractor)
  Vice-President, Software A&E
  Arlington, Virginia

- Mr. J. David E. Gumula (Navy support contractor)
  Senior Staff, Software A&E
  Arlington, Virginia

- Mr. Samuel Levine (Army support contractor)
  Senior Scientist, Computer Science Corp.
  Red Bank, New Jersey

- Dr. Richard J. Sylvester (Air Force support contractor)
  President, P/M Group
  Systems Productivity & Management Corp.
  Dayton, Ohio

EXECUTIVE SUMMARY

Computer software has become an important component of modern weapon systems. It integrates and controls many of the hardware components and provides much of the functional capability of a weapon system. Software has been elevated to this prominent role because of its flexibility to change and relatively low replication cost when compared to hardware. It is the preferred means of adding capability to weapon systems and of reacting quickly to new enemy threats.

Given the prominence of software in DoD's weapon systems, it is imperative that the DoD exploit this important technology. There is a danger that the U.S. military mission can be adversely affected if DoD software development and support is inefficient. There is a perception that such a danger exists and that positive action must be taken to improve the state of software practice in DoD. No plan of action can be effective unless the DoD problems in exploiting software are well understood. Therefore, the DoD Joint Service Task Force on Software Problems was formed at the direction of the Deputy Under Secretary of Defense for Research and Advanced Technology to identify the problems and the opportunities posed by the use of software in computers embedded in weapon systems.

The task force was composed of DoD representatives with broad experience covering most aspects of DoD software and systems development and support. Recognizing the complexity of the issues to be addressed and the limited time to complete the task (60 days), there was no attempt to research new issues and produce a definitive study. Rather, the task force reviewed the numerous reports written by other study groups, relied heavily on the experience of its membership, and sought counsel with others who are currently involved in the development or support of software for the DoD.

This report reinforces the view that there are many difficulties facing DoD in software. These difficulties span the acquisition process, the development and support environment, characteristics of deployed software, and computer professional resources. The difficulties are categorized in an extensive but not all-inclusive problem taxonomy. Many examples are cited illustrating that the problems are severely affecting the development cost and deployment schedules of software, as well as the utility of the deployed weapon systems.

The vitality of the technology base for software in DoD, industry, and academia is directly related to continued U.S. military supremacy. It is generally acknowledged that the U.S. is the world leader in computer technology. Loss of this leadership role may cause a similar reversal in military superiority due to the prominent role the embedded computer and its software play in weapon systems. Several countries - Japan, Britain, and France - have announced government-sponsored initiatives which will improve their software technology base. At least one initiative, the Japanese, is aimed at wresting computer technology leadership from the U.S. In the U.S. there is a growing shortage of computer professionals, which is not only impacting the U.S. software production capability, but is causing an erosion of the academic research and education base as professors are being lured into well-paid industry jobs. The result of this trend is a long term erosion of the U.S. software technological and production base.

Based on its review of the problems and their impact on the military mission, the task force concluded the following:

a. software represents an important opportunity in regard to the U.S. military mission,

b. the state of DoD software development is adversely affecting the military mission,

c. the problems in software are many and interrelated precluding a simple solution,

d. technological excellence in software is an important factor in maintaining U.S. military superiority, and

e. DoD must take a leadership role in solving the problems in software and in averting the erosion of the technology base.

The DoD has a number of current initiatives designed to improve the development and support process. The Ada program is an example of a DoD-level initiative that will improve many aspects of embedded computer software. Each of the Services also has several initiatives aimed at improving software development and support. These activities can be expected to yield steady improvements in the technology base and in the ability to manage software, but the DoD needs a substantial improvement in the state-of-the-practice which

will not result from current initiatives alone. Therefore, the task force recommends that a DoD-level software initiative be established for embedded computer software with strong joint Service cooperation in the spirit of the Ada and VHSIC programs. This initiative should address three major areas:

a. refinement of software acquisition and management practices,

b. stimulation of technology research, development, and utilization, and

c. development and use of expertise of technical and management software professionals.

The necessary technology base and management practices cross Service lines and the problems are bigger than any individual Service initiatives currently address. A plan of action should be developed which embodies a broad and bold approach and which includes both academia and industry as willing participants.

Table of Contents

1. Introduction

This report documents the findings of the DoD Joint Service Task Force on Software Problems. The task force was chartered by Dr. Edith W. Martin, Deputy Under Secretary of Defense, Research and Advanced Technology with support from the Assistant Secretary of the Army (RD&A), the Assistant Secretary of the Navy (RE&S), and the Assistant Secretary of the Air Force (RD&L) to review the problems in DoD embedded computer software and to assess the impact these prob'ems have on the U.S. military mission. The task force wa created because of the perceived need to take positi action to improve the state of software practice for systems. Three Defense Science Board studies within past year have identified software development and su rt as critical problems within DoD. No plan of action c be effective unless software problems are well unders.od.

The task force was composed of representatives from the Services and DoD with broad experience covering most aspects of DoD software development and support. In-depth knowledge of software technology R&D, software acquisition, development, support, software engineering practices, standards, tools, and weapon system characteristics, were represented. Several members of the task force have either lead or been members of DoD or Service-level initiatives relating to improving various aspects of software development and support.

Recognizing the complexity of the issues to be addressed and the limited time to complete the task (60 days), there was no attempt to research new issues and produce a definitive study. Rather, the task force reviewed the numerous reports written by other study groups and relied heavily on the experience of its membership. The task force was particularly sensitive to the need for experience from current projects and sought counsel with others who are currently involved in project management of software for the DoD.

Throughout the document, the task force has illustrated specific problems with examples taken from experience or from studies. Use of these examples is not intended as a criticism of a specific program or manager, nor were they chosen as isolated examples. The current state-of-the-practice is generally inadequate for the task, and the individuals involved often have made heroic efforts in spite of the situation.

The task force observed that software is both an opportunity and a serious problem for DoD, and that the problems in software can impact the military mission. The remainder of the Introduction presents the background and establishes the arguments on impact to the military mission. Section 2 highlights software problems and illustrates its impact on the military mission. Detailed problem descriptions, summaries of cited reports, and case studies can be found in Appendices A, B, and C, respectively. Section 3 presents conclusions of the task force regarding the opportunities and the difficulties in DoD software development and support.

The task force noted that DoD has several initiatives dealing with aspects of the software dilemma. These are summarized in Section 4 and described in Appendix D. While these initiatives are important components of a total strategy, the current level of activity is not adequate to ensure that the DoD take full advantage of embedded computer systems. It is recommended that a broader initiative that focuses the efforts of DoD, industry and academia be started to supplement current Service activities. Section 5 presents the recommendations of the task force.

1.1 Background

Computers are an integral part of DoD weapon systems. Virtually every system in the current and planned inventory makes extensive use of computer technology. Computers are integral to our strategic and tactical planning. They control the targeting and flight of missiles; they coordinate and control the sophisticated systems within high performance aircraft; they are at the heart of the defense of carrier battle groups; and they integrate the complex activities of battlefield command. The military power of the United States, as we know it today, is inextricably tied to the digital computer.

Over the past twenty-five years, the weapon system computer has evolved from a role of minor importance to one of major importance. This trend has been accelerated in recent years by the microelectronic revolution which has been steadily improving the cost/performance ratio of computers--cost has dropped substantially relative to performance and capability. The amazing improvement in cost/performance, coupled with the reduction in size, weight, and power requirements of computers, has made it possible to use computers in weapon systems in ways not envisioned five years ago. The improvement has been so

great that embedded computer systems are now a primary means of introducing new capability and sophistication into our weapon systems.

Software for these embedded computers consists of computer programs and computer data, which are integral to weapon system operation or support, sometimes implemented in firmware. Typically, such software has real-time constraints, performing both a component control function and an integration function for the weapon system (such as inter-component communication and control).

In the early uses of embedded computer systems, the functional capability of the weapon system was embodied largely in the hardware (sensors, control devices, etc.) with the computer performing specialized or ancillary functions. As the digital system has been used to control not only the central function but also the inter-system communications, the role of software has shifted from an incidental role to that of essentially defining the system functions. The hardware is simply the means by which those functions are executed. Today, it is necessary to understand the software in order to understand the capabilities and workings of the weapon system.

The cost of software development and support often exceeds that of computer hardware because the cost of computer hardware has continually decreased while the software has increased in function and complexity. On the other hand, major weapon system hardware production costs (replication costs) are still dominant. This has led to the paradox that software is not always managed rigorously even though it is a key ingredient of virtually every modern weapon system. For example, the Navy AEGIS program management encountered difficulties because the work-breakdown structure of work for the AEGIS system did not go into enough detail to provide a sound basis for management of the contractor software development effort. Without such a basis, software cannot be managed effectively.

It is necessary to understand the nature of embedded computer software to understand the magnitude of the challenge faced by DoD. The software controlling some DoD embedded computer applications is among the most complex, human-designed systems in the world. Embedded computer software exhibits characteristics which differ markedly from other types of software. For instance, ADPE software generally uses numerical and alphabetic input and generates

files, printed reports, or displays for a variety of applications on the same computer. Embedded computer software usually uses analog and/or digital input from a variety of sensors and sources, and generates digital control output to complex weapon system components or status information for human use.

By comparison to the typical business applications of computers, DoD embedded computer software is often required to support much more complex functions. In addition, this software is usually designed and developed in parallel with hardware (target machine and other subsystems). This results in a development activity of higher risk than typical business applications. The critical nature of embedded computer software means that reliable performance is a much higher priority than for business applications. Failures in embedded computer software involve time and dollars, but, more significantly, they involve military weapons and possibly the loss of human life. A failure in a business system usually is measured in time and dollars only.

The management approach, design techniques, and development process for DoD software have many similarities to those used for business systems, but the complexity demands a rigor and scope far exceeding that of business systems. To characterize the size and complexity of embedded computer software from another point of view, it is useful to consider its cost. The software development costs for several weapon systems, including training devices and automatic test equipment, has exceeded $100 million, e.g., B-1B, E-3A, AEGIS, and Safeguard. Such development projects require hundreds of people applied over a period as long as five to ten years. The magnitude of these efforts ranks DoD embedded computer software among the most complex endeavors undertaken anywhere.

1.2 The Relation of Software to the U.S. Military Mission

By exploiting the flexibility of software in development of its modern weapon systems, DoD has elevated the importance of embedded computer software. A function embodied in software is modified for less cost and in less time to meet new threats than if the comparable function were embodied in hardware. The Air Force F-111 program illustrates this point. The table below compares similar capabilities (additional offset aim pointer and updated

weapon ballistics) implemented through hardware on the F-111 A/E and in software on the F-111 D/F. The savings in dollars and deployment lead time in the digital F-111 D/F are striking. Given an existing software support facility, the savings due to making the changes via software rather than hardware have ratios of about 50:1 in cost and 3:1 in time.

Modification	Via Hardware	Via Software	Cost/Time Ratios
#1	$5.28M/42 Mo.	$0.10M/16 Mo.	52.8:1/2.6:1
#2	$1.05M/36 Mo.	$0.02M/10 Mo.	52.5:1/3.6:1
#3	$8.00M/78 Mo.	$0.02M/15 Mo.	400:1/5.2:1

Another example of the advantages which can be derived from changing software without a physical change to the hardware was the reprogramming of the Minuteman III missile.[1] By modifying the software, engineers were able to improve the accuracy of the system without expensive hardware change. The change was designed and implemented in all Minuteman III missiles for approximately $4 million, a relatively low cost for the performance improvement.

In a more recent example, the Sea Wolf missiles, which were just coming into service with the British fleet when the Falklands conflict began, had problems with their control and guidance systems caused by the missile exhaust plume interfering with the television guidance system. This problem was corrected by modifying the software while the ships carrying the missiles were at sea and in combat.[2] The software change enabled the missile to fly offset from the direct line of sight until near the target. The Sea Wolf missiles are credited with six aircraft shot down and two more downed when they crashed while taking evasive maneuvers.

The relative flexibility of software has made it an important factor in modern weapon systems, but that same flexibility puts software in a position to impact the military mission ad-

---

[1] Science, 22 Sept 78, "Technology Creep and Arms Race: ICBM Problem a Sleeper."
[2] Aviation Week and Space Technology, 19 July 1982, "Air Defense Missiles Limited Tactics of Argentine Aircraft."

versely if it is not managed properly. This impact manifests itself as cost escalation, lengthened deployment lead time, and operationally ineffective systems. There is evidence that software is having such an impact and that the impact is becoming more severe with time.

The total cost of software in DoD today has been estimated by the Electronics Industries Association at $5 to $6 billion.[3] By 1990, they estimate that the software cost (including inflation) could reach $32 billion. Although much of the growth is due to increased function and greater use of computers, a cost growth of this magnitude warrants more focused DoD attention.

Another aspect of the cost issue is that software costs are largely labor related. The total labor resource of computer professionals in the U.S. today is insufficient to meet the national demand and the problem is getting worse. The Navy estimates that the software workload for their embedded systems grew exponentially from 1956 to 1978 increasing 150 times in that period. On-board memory of computers in military aircraft over the same period grew from less than 50,000 to almost 400,000 words of memory.[4] This exponential growth will be dampened significantly by the computer professional resource limitations. The result will be postponement or outright elimination of required weapon systems function for new or modified weapon systems.

Software is often the critical path item in weapon system deployment because many of the requirements and design changes that occur during development are absorbed by the software. It is also the integrating element of the system. This again is a consequence of the flexibility of software. Therefore, the efficiency of the software development process and the mechanisms to react to change are directly related to the deployment lead time. If the years it takes to develop a software system can be reduced to months, and if the months it takes to implement major revisions can be reduced to weeks, the U.S. will significantly increase its ability to react to new threats or to pose unanswered threats to our adversaries.

---

[3] "DoD Digital Data Processing Study - A Ten-Year Forecast," October, 1980, Electronic Industries Association Government Division

[4] Words of memory is a rough estimate of the number of instructions needed to tell the weapon system computer what function to perform and how to perform it.

The operational effectiveness of today's weapon systems are of critical concern. The degree to which the embedded systems help meet the operational need and perform reliably is directly related to the quality of the software. Recent experiences where systems have not met the need, or have been faulty due to software, underline the importance of learning how to better develop complex software. This is especially true where software relates to the control and delivery of nuclear weapons.

The impact of software on the cost, deployment lead time, and operational effectiveness of DoD weapon systems is critical. The military mission will be adversely affected unless significant improvement can be made in the state of DoD software development and support.

1.3 The Threat to U.S. Leadership

It is generally thought that the United States holds a position of leadership in computer technology,[5] but this lead can quickly vanish much as the steel and automobile industries' leads vanished during the last decade. There would be a significant difference in the case of computer technolgy. The automobile industry plays an enormous role in the U.S. economy; it does not play the same fundamental role in the U.S. technological lead in weaponry that computer technolgy does. To maintain the lead in computer technology (and by implication, in military supremacy), the United States must not only continue an aggressive hardware technology research program but must also insure the vitality of the software industrial and academic research base, and the expansion of the industrial production base for software. The U.S. could maintain a lead in hardware technology yet lose the advantage by falling behind in software technology.

There is reason for concern that the U.S. could fall behind in this critical technology. Several countries have announced national initiatives. For example:

a. The Japanese government, as a matter of economic policy, is actively promoting the development of knowledge-intensive industries. A specific objective of the Japanese in the 1980's is to leapfrog

---

[5]See "Software Design: Breaking the Bottleneck", IEEE Spectrum, March 1982.

U.S. computer technology to become the world's leading supplier of advanced computer systems. Following two years of study and research, the Japanese have produced a body of ideas and plans for an initiative which they believe will result in "Fifth-Generation Computer Systems" by 1990.[6] A major aspect of this initiative is the concern for software.

b. The French government has established an institution called World Center for Computer Science and Human Resources.[7] The mission of this center is to "unite the social sciences with computer technologies at a rate of development which exceeds that of automation." The individuals chosen to head this center include leading world scientists (several of whom are from the U.S.) , a Nobel prize winner, and several French cabinet ministers.

c. The United Kingdom is focusing on software technology research and development, creating two independent activities. One is sponsored by the Science and Engineering Research Council; it is entertaining proposals from universities to undertake a technically focused effort in software technology research. The other, sponsored by the Ministry of Defense, is focusing on the development of tools and integrated environments.

Although the DoD has a vigorous Science and Technology program, the U.S. has no activity comparable to these three foreign government initiatives, which could ensure the U.S. a position of continued leadership.

The growing shortage of qualified computer professionals is a critical problem in the U.S. Unless attention is given to this shortage, the software development and support resources of the U.S. will not be capable of meeting the demand. This will cause a serious impact to the U.S. economy and will impede the ability of DoD to maintain its superiority in weaponry.

---

[6]"Proceedings of the International Conference on the Fifth Generation Computer Systems," Tokoyo, Japan, October 19-22, 1981.
[6]"Japan's Strategy for the 80's, Business Week, December 14, 1981.
[7]"French World CPU Science Center Stirs House Panel Concern," Electronic News, 7 June 1982.

The academic base for software research is endangered by the heavy demand for software professionals. Professors and new Ph.D.s are leaving the universities for lucrative positions in industry. This trend not only erodes the U.S. software research base, it also hampers the education of system engineers, software engineers, and computer scientists in the numbers necessary to meet the demand. Industry can't and won't solve this problem alone. Although it may be in the best interest of industry to strengthen the universities so that more software professionals are educated, industry is under competitive pressure to hire qualified software professionals from any available source and recently has been choosing the short term expediency of luring university professors to well-paid industrial positions.[8]

Further, industry-based research does not always improve the general state-of-the-art. Competitive pressures often motivate industry to retain their research results as proprietary. Research is usually published or sold only if some benefit accrues to the company. This proprietary approach makes it impossible for one company to build on the innovations of others in any meaningful sense. As a consequence, the software industry does not display the same style of value-added products as is exhibited in other industies.

In summary, the U.S. lead in computer technology is fundamental to U.S. military superiority and the U.S. may be in jeopardy of falling behind in software technology. Other countries have started positive initiatives in software technology. DoD must assure an aggressive leadership role and establish a vigorous program of cooperation among DoD, industry, and academia to ensure U.S. excellence in software and to exploit the full advantages of computer technology for military systems.

---

[8]Communications of the ACM, June 1981, Vol. 24, No. 6, "Eating Our Seed Corn," by Peter J. Denning.

2. Fundamental Difficulties with Software

Software is a critical element in DoD weapon systems and is essential to the DoD mission. It presents enormous opportunities, but the DoD faces a broad range of significant difficulties associated with the development, support, and management of software. There is no single description of a software problem that best characterizes the situation because the difficulties are numerous and interrelated. For example, poor design decisions are often related to management and engineering inexperience as well as inadequate modeling tools to evaluate the design; these bad design decisions may, in turn, result in software that is costly to develop and support; costly support breeds other problems such as a drain on development resource. The multiplicity of interrelated problems makes improvement in software development difficult unless many of the problems are addressed by a coordinated effort. Simply improving the tools of software production will not have the same impact as parallel improvements in the associated software engineering disciplines, and in the skills of the professional staff.

Appendix A contains a description of many of the difficulties DoD experiences with software. The appendix is not intended as a complete taxonomy but rather as a strong indication of the extent of the difficulties. The following sections parallel the taxonomy in Appendix A, summarizing the problems presented in it and illustrating the impact on the military mission with factual examples, where possible. Many of the examples are excerpts from the case studies presented in Appendix C.

Although there are many ways to organize a problem taxonomy, the task force chose four major categories: the life cycle of software, the software production environment, the software product, and software technical and management professionals. These categories represent different and overlapping views of software problems. Several different views are necessary to describe the complex state of DoD software.

2.1 The Life Cycle of Software

The life cycle of software is the set of activities associated with the development, in-service support, and operation of the software from its conception to its retirement from use. The part of the life cycle pertaining to development and in-service support is usually presented in terms of a process model. There are many models of the life cycle process which

differ primarily in the definition of activities and in the relations among these activities. No specific model is presented in this report.

The following sections present a discussion of life cycle problems categorized as requirements, management, acquisition, product assurance, and transition.

2.1.1 Requirements Definition and Analysis

The requirements definition phase of the system life cycle is the least formalized phase in the sense of software engineering disciplines. During this phase system requirements are stated, defended, and negotiated among the users, the development or support agent, and the acquisition agent. This negotiation process relates the estimate of resources needed for the development project to the severity of the threat and the scope of the needs to which the system is directed. Understanding among users, acquirers, and contractors is extremely important. One problem in understanding stems from the lack of precision of English prose. Another is the process of relating the system requirements to lower level software and hardware subsystems. The inclusion of interoperations with other equipment adds to the difficulty of achieving and maintaining understanding among the participants.

Requirements analysis and definition is one of the most critical yet troublesome activities in the software life cycle. The software requirements are often ill-defined and, more importantly, they are characterized by continual change. This is true more for software than other parts of the weapon system because software is the most adaptable to change. Therefore, many weapon system requirements changes are manifested in software changes. The mechanisms to manage this change are inadequate in many projects. In particular, it is often difficult to estimate the impact of a requirements change, adding to the unpredictablilty of software development resources and to frequent cost or schedule overruns.

Not only are some embedded software requirements difficult to define and highly volatile, but they have proven to be difficult to validate. This results in deployed systems that do not meet the full need of the user. The COMPREP System experience is an example of this situation (See Appendix B.8). This message processing system was abandoned after development when

it failed to pass its operational evaluation because it was deemed inappropriate to the operational needs of its user. Military units had to operate without a needed message processing system; this impact to DoD perhaps was more significant than the lost dollars.

Too little is known about how to assess the impact of a requirement, or a change in requirements, on software size and complexity. This problem exists not only for the mission system but also for its support systems. For example, small changes in aircraft performance requirements may lead to large changes in the air crew training device software.

On the positive side, properly designed software can be advantageous in handling changing system requirements. The addition of relatively small amounts of software can have a large effect on system success. An excellent example of this is found in NASA's Viking program, which was the mission to Mars. Originally, the ground support software was to receive and process simultaneously, the data from both Mars orbiters. The development contractor (after development was well along) recommended that, since each orbiter had on-board data storage, alternating (rather than simultaneous) transmission and processing be permitted. This change saved an estimated 30% of the already high projected costs of software without degrading the mission. Another issue on Viking was the ability to transmit new computer programs to the space vehicles. This requirement was added at relatively low cost and proved critical to achieving the launch date and extending the mission life of the spacecraft.

2.1.2 Management of Life Cycle Activities

Management of software life cycle activities exhibits many associated difficulties. Planning is often inadequate. This is due in part to the difficulty of estimating development resources accurately and of evaluating the impact of changes in requirements. These planning and control difficulties place management in a position of reacting to situations after they occur rather than anticipating them in time to prevent their occurrence or minimize their impact. The result is both added cost and extension of the development schedule.

Managers possessing little understanding of embedded computer software are often thrust into positions where their decisions directly affect the software process. This can be

manifested in many ways including: inappropriate contracts, lack of strong systems engineering in software development or modification, lack of attention to disciplines that are appropriate to software development, lack of effective plans for software activities, inappropriate methods and tools for tracking software projects, inappropriate assignment of responsibility, and inappropriate prediction, establishment, and monitoring of schedules and budgets.

Part of the difficulty in improving our ability to manage software life cycle activities is the lack of suitable measurement data or metrics. Without this data, management will continue to be hampered in planning, controlling, and evaluating software efforts. Measurement data on past projects are a prerequisite to adequate estimation of new projects. Measurements throughout the life cycle are necessary to give management the information required for control. Measurements are needed to allow management to evaluate new techniques. The current state-of-the-art does not permit software professionals to quantify the benefits of techniques because little empirical data exists.

Acquisition and engineering managers lacking familiarity with embedded computer software are often thrust into positions of responsibility. Some managers ignore the software development and support aspects of their jobs. Officers with little acquisition and embedded computer software experience are often responsible for projects involving millions of dollars. Fixed price contracts have been used inappropriately for high risk development causing financial strain to the contractor and cut corners on the developed software.

There is a lack of appreciation for discipline and good communication among systems, specialty, and software engineers. There is a need for measurable milestones and planning, assessment of quality, and planning for integration and test of software and systems.

There are no generally accepted models that can predict development size, complexity, staffing, schedule and budget for software. Unrealistic schedules, difficulty in monitoring development, and poor planning for the life cycle remain problems for some service elements. There is often a lack of good management discipline in all life cycle phases.

Foreign Military Sales (FMS) aggravate issues concerning release since software can contain intelligence, technology, performance and vulnerability information. Lack of understanding at the top government levels leads to inconsistencies in guidance and split responsibilities. Vague and incomplete disclosure letters leave the services in the awkward position of trying to satisfy conflicting DoD and State Department guidance. FMS also aggravate the design and skill shortage problems in the software area.

2.1.3 Software Acquisition

There is a lack of uniform methodologies for handling the acquisition of software. The detailed methodologies for hardware contracts are often inappropriate to software but are often used. Interfaces are poorly maintained with weak configuration control when different parts of a system are developed by different contractors because focus is normally on hardware with too little attention to software. Documentation is often of little value in understanding project status, because it is either absent or late. Procedures for change control during the procurement process are often time-consuming, inconsistent and ineffective. The procurement process is so lengthy that changes to requirements occur during the process. This causes further delay and, if uncontrolled, these changes can cause unnecessary cost escalation.

There is a great reliance within DoD on software contractors for creating the software. Contractors often convince project managers to permit use of non-standard techniques, tools and hardware which do not always offer an overall life cycle benefit in that they focus on the development cycle and totally ignore or pay only lip service to the in-service support.

A frequent error for embedded computer software is the belief that software and hardware can be developed and supported by different companies or agencies without a strong systems engineering authority to allocate requirements, coordinate design, resolve conflicts, and perform the system integration function. One particular C^2 system, started in 1973, attempted acquisition with separate hardware and software contractors. The software development cost went from $10M to $40M due to requirements changes and configuration differences. No contractor was responsible for overall system delivery. The program was cancelled in 1978 because the software would not work properly with the hardware.

2.1.4 Software Product Assurance

Software product assurance includes all the life cycle activities associated with demonstrating the correctness of the software product (e.g. requirements, design, code) and its quality. This is done through analysis, reviews, and testing techniques.

Product assurance of software through testing should be a continuing process throughout the system life cycle. This requires allocation of adequate resources for test planning and cooperation among the acquisition agent, development contractor, test agent, support agent and user. Independent verification and validation (IV&V) implemented early in the system life cycle can help achieve better quality software.

Too little is known about different testing techniques and how much testing is enough. Secure systems (not just operating systems), artificial intelligence systems, and distributed systems are applications where there are no good criteria for various applications to determine how much testing is optimal at each step in the software development process. In a recent digital flight control system development, during software integration, there were so many hard-to-find errors (due to insufficient module testing) that the integration was curtailed and more extensive module testing resumed. This caused a substantial slip in schedule.

Test environments are different from the real environment - a factor that influences the amount of test and the software reliability. Too often, testing is governed by the amount of time that is available. Optimal test requirements for different embedded computer software applications need far better definition.

2.1.5 Transitions in the Life Cycle

There are several points in the life cycle of software where problems occur in the transition to succeeding activities. Two of the more difficult transitions are from exploratory research to engineering development and from development to support. The former is characterized by new and rapidly changing technology, while the latter requires a stable technology base.

The very rapid change in computer technology has a destabilizing effect on acquisition and support engineering. For instance, firmware is widely used in today's weapon systems yet

acquisition familiarity, policy and support concepts have not kept pace with the technology. Contractual requirements intended for software have been disputed by contractors as not being applicable to firmware. For engineering development, the economies of hardware and software are shifting, software becoming the dominant cost item, but engineering tradeoffs are still imposed which favor hardware optimization. The DoD Acquisition Improvement Program has provided guidelines toward deciding whether the most recent technology should be incorporated in weapon systems or whether state-of-the-art research and development should be followed with pre-planned product improvement; and how technology transfer should take place using contractor incentives; but it is not clear how these guidelines are to be meaningfully applied to embedded systems software.

The transition from development to in-service support (sometimes referred to as "maintenance") is another such point where difficulties occur. These difficulties are usually characterized by differences in software support environments, standards, and methods between the development group and the in-service support group. There are several cases where this incompatibility has cost DoD millions of dollars.

2.2 The Support Environment

The software support environment is defined as the methods, tools, and facilities used for software development or in-service support. Problems in the support environment are categorized by the lack of disciplined methods, labor intensive process as caused by inadequate tools, reinvention of software, and insufficient capital investment. These four categories are discussed in the following sections.

2.2.1 Disciplined Methods

Software has been developed and supported by contractors (and in-house personnel) who are not required to practice adequate engineering discipline in software activities. This lack of discipline varies from those who recognize the necessity for discipline but fail to apply it on a particular program, to those who fail to recognize software as requiring discipline. Some of the shortcomings include: lack of adherance to good

programming standards, lack of adequate configuration management practices, inadequate baselining and documentation, and lack of adequate system engineering practices applied to a program. Major causes of this lack of discipline are: inadequate development standards, ineffective product assurance programs, inadequate subcontractor management, and deficiencies in contract requirements.

The problem of an inadequate software engineering discipline is characterized by the laissez faire attitude in the selection of techniques used from one project to another, antiquated programming methods, ad hoc requirements analysis and design, and the informal methods of testing complex systems. This situation has an adverse effect on development costs but can have a more significant effect on in-service support costs. Varied styles of documentation and coding, coupled with inadequate quality due to ad hoc techniques, can significantly increase the cost of maintaining and enhancing deployed software.

2.2.2 Development and Support Tools

Software engineering is, by its very nature, labor intensive. However, this is aggravated by the fact that techniques and tools are out-of-date, insufficient, inefficient, and limited. Automation of mechanical processes is occurring but more is needed to improve productivity. Automated tools could increase productivity but there are no accepted productivity measures for tools or tool sets.

The approach used in developing the F-15 software support facility illustrates the level of success possible in achieving tool uniformity. The F-15 Avionics Integration Support Facility (AISF) was developed by WR-ALC/MMEC, beginning in late 1976. The approach taken for development was to be as common as possible with the existing F-111 facility. This approach was intended to serve two purposes: (1) software would be transportable between facilities; and (2) by keeping the design concept as similar as possible, a model would be available for scoping, pricing, sizing, etc.

The design approach was built around a common support concept with common processors and peripherals, common language (where possible), and other common hardware. Due to the vast difference in weapon system mission and avionics system architecture, only a minimal amount of software was directly trans-

portable. However, much of the previously developed F-111 support software approaches and algorithms were reused. It was the opinion of personnel closely associated with the background of the F-15 facility planning that resource estimates would have been grossly underestimated had they not decided to use the F-111 facility as a model.

Proliferation of languages and instruction set architectures (ISA's) increases development support costs such as training, documentation, and tool reinvention. Personnel are unaware of tools and models that could save time. Tools are either unpublicized, hard to understand, or inefficient. Often, support tools are not useful because they are non-reusable due to unique language/ISA, have poor quality and poor documentation, or consist of a set of tools which do not interface and are not user-friendly. In addition, there is difficulty in making the transition from development to contractor and support personnel of non-weapon system specific tools.

Automated support of the software life cycle process activities is heavily oriented to the single activity of code generation which can represent a relatively small part of the total development effort. Computer-based tools to support requirements analysis, design, verification and validation, and management are inadequate. What tools do exist in these areas are not designed to work with tools supporting other activities. The consequence is that the development process remains heavily labor intensive and prone to errors. With the rising cost of software professionals, continuation of the current situation will become more and more costly to DoD.

2.2.3 Reinvention in Software

A concern in software development today is the inability to make use of functionally similar software developed for other systems. Reuse of software has been limited because of the difficulty of determining if functionally similar software exists, and because the large majority of software is not designed with reuse in mind. The result is a high degree of re-invention in software development with the impact of higher costs.[9]

---

[9]The Air Force does not perceive this concern with equal priority.

There may be abundant opportunities for reusable software components within such areas as realtime executives, file management subsystems, display software and report generators, which, though developed independently, represent very similar functional capabilities.

The elimination of reinvention may represent a large opportunity for cost avoidance for DoD, potentially measuring in the hundreds of millions of dollars. This opportunity is illustrated by the examples presented in Appendix C.3. One of the examples presented, the Navy Command and Control System (Ashore) software, represents an opportunity cost of over $10 million. Although the cost savings potential is significant if reinvention can be eliminated, there are other important payoffs. Namely, development lead time can be reduced and software reliability can be improved.

2.2.4 Capital Investment

In order to solve many of the software problems facing DoD, capital investment is required. To date, software-related capital investment has not been sufficient. The Army's Post-Deployment Software Support (PDSS) Centers illustrate this point. The funds required for capital investment and labor (R&D, OMA, MPA, and MCA funds) for nine support centers is $404 million for the period 1982-1987. The funds available are $135 million or a shortfall of $264 million. This funding shortfall will seriously impact the effective deployment of Army systems.

The area of software support environments represents both a great opportunity and a necessary burden for DoD. Better support environments are a prerequisite for better software engineering and the elimination of reinvention. The investment necessary to attain this (equipment, education, etc.) is large, and there has been an unwillingness to make the necessary capital commitment. What money is allocated to software support is often reprogrammed to other weapon system needs. Software is viewed as a low priority item by management when it comes to a funding crunch. If DoD is to solve its current software problems which pose a serious threat to its military mission, software must be given the proper priority in funding.

A subject related to capital investment is government-funded independent research and development (IR&D) by industry. There are too few good ideas for software research projects

being funded in the IR&D arena. The task force has the perception, which it was unable to validate, that software IR&D projects get low scores because evaluators are not software oriented, and results are not as tangible or quantifiable as the competing projects.

2.3 The Software Product

The software product is defined as the operational embedded computer software and any material required for adequate life cycle support - requirements specifications, design specifications, source code, test data, system generation data, unique support tools, etc. The problems associated with the software product are categorized as not meeting the need, difficult to measure, design attributes, and documentation. These categories are discussed in the following sections.

2.3.1 Software Utility

Deployed software often fails to meet the full need of the intended users. This can occur when the original requirements were incorrectly stated or when requirements were incorrectly implemented. Ambiguous, unclear, and incomplete communication between the user and implementor causes undetected omissions or internal contradictions. In some cases, it is difficult to assess the real user need without first placing a system in the field for the purpose of use and evaluation. Since completion and performance criteria are often inappropriate, incorrect, or impractical, product evaluation is difficult. In summary, the right requirements may not be stated and, even after the embedded computer software has been developed, it may be difficult to decide if the software meets its stated requirements.

An example of requirements uncertainty or conflict is provided by a recent fault isolation concept for a weapon system. The initial concept for fault isolation was to provide a rigid step-by-step guide to maintenance personnel through a video display. Although this was appropriate for inexperienced maintenance personnel to use, there was some concern that experienced maintenance personnel might want to avoid many of the pre-planned steps to get quickly to the tests that would exercise the components suspected to be faulty. An early hands-on evaluation of the system indicated that both approaches were necessary; hence, the software had to be modified to accommodate the additional requirement for ready access to specific tasks.

2.3.2 Software Metrics

Good analytic models and hard empirical data on software are lacking. Without these it is difficult to estimate development resources, evaluate future cost and mission impact of embedded computer-related decisions, or evaluate the positive benefit of new software engineering techniques. Existing software cost estimating models are inappropriate because they require data not available until the project is underway. Experience with these models indicates that they do not provide accurate estimates. To create better models and compute better estimates, a comprehensive data base of measurements is needed. Such a data base is not available today. This is due in part to the reluctance of companies to share data. The lack of a standard set of metrics also hampers collection and sharing.

There are no validated models of life cycle costs and productivities in embedded computer software development and support. For example, difficulty with using current models can be seen in these figures taken from a guidebook on software cost estimating:

Project	Forecast Total MM	Actual Total MM	Ratio of Forecast/Actual
A	419.7	71.0	5.9
B	2288.5	991.7**	2.3
C	51.5	43.8	1.2
D	3298.7	514.8**	6.4
E	7.9	7.3	1.1

(** Contains some estimate-to-complete data, along with actuals.)

One reason that metrics are not well-defined is that there are so many different approaches to software development and support. Until uniform software engineering approaches are defined, it will be difficult to define standards or guidelines for data collection. Such standardization is essential to the automation of the collection and data reduction activities which will make measurement cost effective and practical.

2.3.3 Design Attributes

The design of a system strongly influences its eventual total cost, how long it takes to develop it, and how amenable it is to change during development and support. The concepts and constraints for embedded computer software design are sensitive to technology--especially the architecture and capacity of the computational system (computers, memories, data buses etc.) and the role embedded computer software must play in the weapon system. Embedded computer software design demands a thorough knowledge of many fields--the weapon system, engineering, computers, and software technology. Good design provides many benefits to a system while poor design causes many difficulties.

Many of the problems associated with the software product find their source in design. Poor choices at the system design level can place constraints on the software that result in cost escalation. Software design which does not properly consider future changes increases the life cycle cost and limits the ability to react to new threats through software modification.

In addition, poor human engineering or lack of robustness of the software may make a system operationally unacceptable to the user.

Software's remarkable flexibility for change (when compared to hardware) often masks a serious problem--most embedded computer software cannot be changed as easily as it should because of its inflexible design. This is illustrated by systems where relatively small requirements changes (measured from the point of view of function) impact many aspects of the software and force a disproportionate amount of redesign and code generation. The cost of this inflexibility is significant. More disturbing is the inefficient use of scarce professional resources. They could be applied to the implementation of further enhancements which, otherwise, have to be delayed or cancelled.

The software implications of design decisions made at the weapon system or embedded computer system level are often ignored or misunderstood with disastrous results, as illustrated by the following example. An aircraft engine control system was based on an eight-bit microprocessor architecture. This

choice required the use of double precision in the software to achieve the necessary accuracy. The use of double precision resulted in timing and memory problems. To solve these problems, algorithms were revised and a more complex software design was necessary, severely complicating verification and testing. The project was delayed when errors became increasingly difficult to find and correct. Some might blame the software for this failure when, in fact, the root of the problem was poor decisions at the system design level because personnel did not understand the application.

2.3.4 Documentation

Documentation plays many roles in the development, operation, and support of embedded computer systems. Its focus can be anywhere from the broad concepts of a system design to minute details. The primary role of documentation is to convey information. During development, documentation captures the status of the software system at its current stage of development, including requirements, design, design trade-offs, implementation status, etc. This documentation allows the personnel of a large project to communicate to each other the information necessary to develop and manage a complex system. When the software is deployed, this documentation is used by the in-service support group to make modifications to the software. Modification of large software systems can be very costly without adequate documentation. During operation, documentation on the operation of the software is necessary for effective utilization by the user.

Documentation for embedded computer software is often inadequate for cost effective in-service support. There are a number of reasons for this. Documentation of a complex software system can be voluminous and costly to produce. If the development agent is under cost and schedule pressure, which it usually is, documentation is one of the first items to be cut or deferred. Until software documentation is an integral by-product of the software engineering process, supported by automation, and maintained in an electronic form, poor documentation will be a continuing problem.

When documentation is produced, it very often lacks the important attribute of traceability -- relation of a requirement to the design and code components that implement it. This makes it difficult to evaluate the impact of requirements changes. It also impedes the updating of documentation to reflect the continued changes to the software.

2.4 Embedded Computer Software Professionals

People are the most important resource in any software development or support effort. Design and redesign remains an intensive intellectual activity even after the best of automated tools is applied. Not all software professionals are so adept as others. The variation in effort among different individuals doing the same job can be as much as 25:1. With this degree of variance, it is no wonder successes and failures alike in software can be traced to the skills and experience of the professionals involved, both technical and management. Many rate the capability of the development team as the most important productivity factor.[10]

The problems relating to personnel are categorized as skills, personnel supply and proper incentives. These categories are discussed in the following sections.

2.4.1 Embedded Computer Software Professional Requirements

The key skill required by embedded computer software personnel is the ability to communicate across traditional engineering, computer science, and management disciplines to evolve coherent software requirements, designs, and modifications. Lack of currency for both technical and management personnel impedes the low risk transfer of new technologies to weapon systems. The government needs mature and adequate technical and management skills to prepare requirements, monitor developments, and support future changes. Acquisition and support skills for embedded computer systems are not taught in universities.

Qualified managers and professionals must have a wide range of skills and experience. The example in Appendix C.3 illustrates the difficulties that can be caused by inexperienced management. The management decisions concerning the signal processing computer and proper reserves for immature GFE/GFI caused a three year delay and millions of dollars in cost overruns. The ultimate impact was a three year delay in the deployment of a major surveillance system.

---

[10] See Software Engineering Economics by Barry W. Boehm, Prentice Hall, Inc., 1981.

The personnel problem is exacerbated by the limitation of most entry level and middle technical/management civil service positions to the Engineering (GS-800) series in the commands that acquire ECS. This excludes computer science and other related degree fields from pursuing careers or shifting to careers involving ECS acquisition. It should be noted that Civil Service regulations currently prohibit advertising a position as interdisciplinary when one of the disciplines is a "Professional" series (as is the GS-800 Engineering Series).

2.4.2 Availability of Qualified Professionals

The demand for software engineers and computer scientists exceeds the supply. There are no data on which to determine the true need, particularly for software acquisition personnel. The Service policies of rotating officers through a variety of jobs reduces and disrupts the supply of qualified personnel. For all Services there is a lack of adequately trained civilian and military personnel to satisfy post-deployment software support requirements. The shortage has caused the Services to become heavily dependent on contractors. This reliance creates problems with personnel turnovers, long learning curves, and less skilled personnel.

The Navy FCDSSAs are a good example of the projected impact of personnel shortages. By 1985, the projected shortfall in professional software personnel (as compared to demand at the FCDSSA in San Diego) will be over 200 people, and by 1990 it will be over 600 which means less than 50% of the demand will be met by the available resource. (See Appendix C.2)

The Army Post-Deployment Software Support Centers estimate a current shortage of 300 civilian and military personnel required for the manning of nine centers. This shortfall reflects a spending shortfall for the centers, but even if the funding were available, it is doubtful that the required personnel could be obtained.

The competition for qualified personnel will intensify with the increasing shortage in computer professionals. DoD will have difficulty competing with industry. For instance, the COMTEC 2000 study (Appendix B) shows that the second term retention rate of enlisted personnel with certain computer resource skills is only 50% because of the attraction of industry jobs.

The Electronic Warfare Support Directorate at Warner-Robins Air Logistic Command is another example of the people problem. They (WR-ALC/MMR) are only 60% staffed against their engineering slots. This shortage is aggravated by Foreign Military Sales of the Electronic Warfare systems they support. Since November, 1981, there has been a 30% increase in such sales (FMS EW cases). This number is expected to grow since there are more sales being negotiated. Though this activity is strongly driven by the policies of the present U.S. administration, it results in the requirement for long term support. Staffing the FMS work is difficult at best and tends to drain skilled people from U.S. embedded computer software work.

The problem of shortages in qualified software professionals will not be solved easily. This problem is not new, but it has not been perceived to be a pressing issue by DoD. As more demand is made of the current limited resource, corners will be cut, gradually reducing the operational capability and reliability of our weapon systems until failures become painfully more frequent.

2.4.3 Incentives

Present incentives favor migration of skilled personnel from job to job. Government software talent moves to other government jobs or to industry for promotion, better working conditions, and educational opportunities. Perceived salary shortcomings reduce the government ability to attract highly qualified people. There is no formal, effective career management program with classification/qualification standard adjustments, continuing professional education, challenging assignments, and identification/communication of career paths.

The following career pattern for entry level software system engineers has been observed at one Navy field activity. Entry level personnel are difficult to obtain. For those engineers that enter at GS-7 level, promotion to GS-9 occurs in one to two years. At this point, the software engineer is faced with a career decision: to market his valuable skill to the private sector or remain within the civil service. Approximately 60% opt for the private sector. Those electing to remain in the civil service generally reach GS-12 level in two to three more years. Then even the best are faced with a career stop causing another decision at which 40%, usually the top achievers, turn to the private sector. This loss of mature journeyman software engineers is the most difficult for the organization to absorb.

There needs to be a "software engineering" career field for both military and civilians which recognizes the skills needed are a combination of engineering and computer science. The career path should encourage the development of managerial and technical skills and experience pertinent to embedded computer software applications.

3. Conclusions

The improvement in the cost/performance ratio of digital computers over the past 25 years, along with the reduction in the size, weight, and power requirements of digital hardware, has resulted in an increase in the use of digital electronics in weapon systems. Today, most weapon systems are dependent on digital systems and embedded digital computers. Since software is inherently more flexible than hardware, software has become the means of capturing the essential "intelligence" and control functions of the weapon systems, making the systems more adaptable to changing threats. Given the important role software plays in weapon systems, problems in the development, support, and operational effectiveness of software can adversely impact the military mission. Conversely, improvement of the process and of the quality of the product can have a positive impact on the military mission.

The Joint Service Task Force on Software has reviewed the problems in embedded computer software and the impact these problems are having on the military mission. The conclusions of the task force are:

a. Software represents an important opportunity in regard to the U.S. military mission. Digital computer software provides the means to incorporate added capability in weapon systems. Its characteristic of ease of change in comparison to hardware provides an invaluable means of quickly reacting to new threats.

b. Technological excellence in software is an important factor in maintaining military superiority. Initiatives by the governments of other countries are aimed at the attainment of computer and/or software leadership for these countries. The U.S. has no comparable initiative. In addition, economic forces coupled with the shortage of computer professionals threatens to erode the academic base for education and R&D.

c. The state of DoD software development and support is adversely affecting the military mission. The problems in software development and support are many and varied. They combine to cause cost escalation and deployment delays. The net result

is unnecessary lengthening of the lead time required to meet new enemy threats. Function is cut and quality suffers as the demand for new systems increases. The result can be an ineffective or faulty system. If improvements are not made, it is reasonable to expect more failures in operational weapon systems in the near future.

d. There is no single "software problem." Appendix A presents an extensive list of problems in software faced by DoD. It is necessary to solve many of these problems in order to solve one for they are interrelated. Therefore, any initiative addressing DoD software problems must be broadly based to be effective.

e. DoD must take a leadership role in solving the problems in software and in averting the erosion of the technology base. DoD cannot rely solely on industry or other agents to solve the problems it faces in software, or to assure the health of the software technological base. DoD must take the initiative to support existing Service programs and to establish programs that will join industry and academia with it in solving the problems while continuing to stimulate R&D.

4. Existing Initiatives

The DoD has a number of current initiatives designed to improve the efficiency of the development and support process and the quality of software. The breadth and complexity of DoD's software causes a wide variety of problems. For this reason, DoD's initiatives range widely in purpose and scope.

At the joint Service level, the Ada and Joint Logistics Commanders' (JLC) initiatives are the most prominent. Others involve software quality assurance and testing.

Each Service has a number of initiatives; only some are mentioned here. The Air Force Systems Command has an integrated research, development, and acquisition planning system, entitled Vanguard, to integrate technology base activities into appropriate mission area and functional plans, including their Computer Resource Functional Plan. In addition, the Air Force Logistics Command has begun an Embedded Computer Systems Support Improvement Program (ESIP). The Army Post-Deployment Software Support (PDSS) initiative focuses on its battlefield systems--how to provide effective and economic support while ensuring a close working relationship between the combat user and the material developer. The Navy has a set of complementary initiatives: the Tactical Embedded Computer Program Office (TECPO) is a project office for coordinating the development of all embedded computer resources within the Navy; the Software Engineering Environment Working Group (SEEWG) program defining the Navy's strategy for definition, development, and introduction of a Navy standard software engineering environment; a systems command program for coordinating all Program 6.2 advanced development funds; and a study to determine the full scope of the Navy's embedded computer system products. Details of these and other activities are found in Appendix D.

The DoD maintains a close relationship with academia and industry. DoD support for academic research provides an infusion of ideas and is one of the cornerstones of technology. Industry has the challenging (and difficult) task of developing DoD's requirements into hardware and software, often at the leading edge of technology. Thus, DoD has a deep and critical interest in the technical vitality of industry and academia.

These activities can be expected to yield steady improvements in the technology base and in the ability to manage software. But, a substantial improvement in the state-of-the-practice is needed. Such a jump will require a coherent plan for focusing technology and non-technology initiatives. DoD now has many initiatives but they need to be bolder, broader in scope, better coordinated, and involve academia and industry.

Support for these initiatives must be from the level of the Secretary of Defense. This gives them the impact that cannot be felt by initiatives at lower administrative levels.

5. Recommendations

The DoD Joint Service Task Force on Software recommends that a DoD-level software initiative be established for embedded computer systems with strong joint Service cooperation in the spirit of the Ada and VHSIC programs. This initiative should supplement, integrate and build upon existing DoD and Services' activities.

Such an initiative would:

a. provide visibility and credibility to the Services' articulation of the importance of software in the military mission, and of the problems currently realized in software,

b. serve as a forum for coordination among individual Service initiatives and on-going joint efforts,

c. serve as a focal point for technology initiatives relating to software within DoD, and

d. address some of the manpower, educational and productivity issues associated with software development and support.

In order to define and direct the initiative properly, the task force recommends that a plan of action be developed. The plan should present the basic DoD strategy on software. The plan should be developed with an understanding of the role software plays, the current problems and their related impacts, and important current Service and DoD initiatives.

It is recommended that each Service continue its individual efforts to characterize and improve its software environments, to highlight and solve problems and to investigate and pursue areas of significant pay-off. The plan should be refined as necessary based on the on-going Service efforts.

The Task Force recommends that the DoD software initiative address three major areas: acquisition/management, technology, and expertise. These three areas are defined below and are partially elaborated by specific recommendations. The list of specific recommendations is not meant to be all-inclusive. Rather, it is a representative starting point for the Plan.

a. Refine and support software acquisition and support management practices: Incentives must be defined to encourage quality products with reasonable development and support costs. Techniques for effective management of all life cycle activities must be developed. Some specific recommendations are:

(1) Adequate Monitoring - Contract monitors have no adequate way to measure progress during software development and to measure acceptability of the software product. Consequently, the contract monitor seldom has the technical or management basis to withhold payment or to negotiate. Better monitoring approaches or tools should be developed.

(2) Life Cycle Support Requirements - The DoD should establish a quantitative life-cycle model which recognizes that changes to software are going to occur. This model must reflect interactions within the development cycle, redefining requirements, revising specifications, redesigning, changing code, retesting, and preplanned product improvement. Support requirements, contracting mechanisms, procurement strategies, and personnel selection must be oriented to the flexibility of software to support rapid controlled change where required.

(3) Incentives - Contracting incentives should encourage development of quality software. Incentives which would encourage the reuse of existing software should be instituted. These incentives should be tied to useful and meaningful metrics.

(4) Microprocessor/firmware Policies - The DoD should formulate uniform policy governing microprocessor and firmware definitions, identification/labelling, concept of operations, configuration management, higher order languages, and data item descriptions.

b. Stimulate technology research, development, and utilization: The proper circumstances must be created to allow one research group in government, industry, or academia to build on the breakthroughs of others. This must include an effective means of quickly moving promising research products into evaluation and practical use. Some specific recommendations in this area are:

(1) Impact of Requirements on Systems - Mechanisms must be developed to assess the impact that requirements and requirements changes have on the complexity of software and systems. Historical information should be consolidated, and models should be developed. Metrics and tools need to be developed to assist software engineers in assessing the impact of proposed changes to the system.

(2) System Design - Research into designs and design practices for a variety of system types should be supported. System and software design for reuse, distributed computational architectures, highly parallel processing, design for requirements change, design for minimum errors, design for testability, design for support and pre-planned product improvement, and design techniques in the event of computer plenty should be developed and evaluated. VLSI development will yield substantially reduced hardware cost, thus designs consistent with new economic and performance factors must be pursued.

(3) Empirical Data and Metrics - Uniform metrics and measurement aids must be developed to support collection of consistent and useful cost estimation, productivity and quality data. The data should then be used to develop cost prediction models, and provide the foundation for developing and evaluating analytical automated support tools and practices.

(4) Common Tools - An integrated environment, which implements consistent software development and support methodologies should be constructed. Experimentation is needed to demonstrate the utility of the methodology. Metrics are needed to evaluate the productivity of the tools.

(5) Support for Documentation - Automated assistance in the preparation and maintenance of documentation is needed. Methods which more closely relate documentation to the code and which ensure greater consistency and currency are needed. Examples of various documents for different uses and different reading audiences should be generated. It is appropriate to define the levels and type of documentation required to support effective ownership of different types of software. For example, research which would support the capture of graphic representations of designs could yield effective tools to support modification. This effort should build on present JLC efforts to define Data Item Descriptions (DID) to bring DoD data items in line with the current state-of-the-practice and should complement follow-on activities to address generation and documentation of data items by advancement of software engineering techniques and automated tools.

(6) Rapid Change - The DoD needs to develop a better strategy for dealing with high technology areas which experience continuous and rapid change. Recognizing that software often supports systems in which the requirements or the technology are likely to undergo change, development of techniques which efficiently accommodate change is needed.

(7) Technology Communication - Greater communication should be promoted among Service elements developing similar technology. In common software technology areas, lead elements could be designated to ensure that a critical mass of

research exists. Care must be exercised not to prevent other Services from continuing work in that area so that the technology communication and insertion across the Services and throughout industry not be hampered.

(8) Technology Evaluation and Infusion - A framework for experimentation and evaluation should be defined to enable selection of evolving software technologies and to guide the insertion of new technology into DoD projects at low risk to those projects.

(9) IR&D - Qualified DoD software personnel should become more involved in review of industry IR&D activities involving software. The IR&D rules should provide appropriate incentives to encourage companies to pursue software research.

c. Develop and use expertise: Comprehensive programs to increase the skills of the current technical and management software population must be started in government and encouraged in industry. University computer sciences and software engineering curricula must be improved and capacity expanded. Some specific recommendations are:

(1) Qualified People - The DoD should define and advocate systems and software engineering career fields. It is essential that the DoD establish a program to address the education, training, recruitment and retention of systems and software engineers and managers. Career progression for both civilian and military need to be defined and instituted. For instance, the West Coast Region of the Professional Council of Federal Scientist and Engineers conducted a study which recommended a civilian software engineering series; the series has been delayed by the Office of Personnel Management. That effort should be revitalized and given high level support.[11]

---

[11] In addition, the Naval Material Command Office of Personnel Management is hosting a conference in October to define software engineering, discuss personnel problems, and develop a plan to improve problems relating to software engineering personnel.

(2) Government/Industry Exchange - A program encouraging exchange of software personnel between government and industry should be initiated. Care needs to be exercised that the exchange uses competent people and is not viewed as training for the inexperienced.

(3) Productivity Measurement - Measures must be established for evaluating productivity of programmers, software engineers, and managers, and for relating appropriate skills to the quality of the product.

Summary: Problems exist in the design, procurement and support of software systems. These problems cannot be ignored. They need a focused, coherent, market-oriented strategy at the DoD level to ensure solution. The problem is bigger than any individual Service can address alone and needs a broad, bold approach, which includes both academia and industry as willing participants.

APPENDIX A

Software Problem Taxonomy and Descriptions of Software Problems

INTRODUCTION

Appendix A describes problems in software. They are structured into four major categories and sixteen minor ones. Each Service described the most critical software problems experienced by that Service, although not all problems were experienced by all Services. The result is a set of problem descriptions, written by the Services and melded into one set of software problem descriptions.

APPENDIX A: PROBLEM TAXONOMY

A. Life Cycle

The term "life cycle" includes all activities (including the relations among activities) involved in the development, operation and support of embedded computer software from its inception to its retirement. Problems included as part of the life cycle were divided into the five major subcategories of requirements, management, acquisition, product assurance and transition. The problems described under life cycle are some of the most ill-defined and difficult problems to bound or to solve. They involve working with and through people on items or processes that are extremely complex, as well as often technically demanding.

PROBLEM: A.1 Life Cycle: Requirements

DESCRIPTION:

Establishing Requirements - System requirements are usually negotiated among the user, the supporter, and the acquisition agent to meet an enemy threat or to take advantage of new technology within the confines of financial resources and schedule factors. It is essential that communication be free and adequate to ensure that no misunderstanding and, hence, subsequent conflicts arise. This process is sometimes impeded by disagreements, and sometimes by uncertainty on some of the requirements among the parties.

Conflict is particularly true when the system to be developed is first-of-a-kind. Only after some experience on the project can the user or acquisition agent begin to see what is truly required. Here again communication is important.

An example of requirements uncertainty is provided by a recent fault isolation concept for a weapon system. The initial concept for fault isolation was to provide a rigid step-by-step guide to maintenance personnel through a cathode ray tube display. This was designed for inexperienced maintenance personnel to use. There was some concern that experienced maintenance personnel might want to avoid many of the sequential steps to get quickly to the tests that would exercise the components they suspected were faulty. An early hands-on evaluation of the system indicated that both approaches were necessary; hence, the software had to be modified to accommodate the additional requirement. It is not clear that this requirement could have been defined earlier in the program.

Communication with contractors is also extremely important to ensure that requirements are clearly conveyed. Differences in understanding and interpretation are common and are often reflected in the software implementation.

Using English Prose for Requirements - The inadequacy of the written word is a subtle, but important, problem. Requirement statements tend to use the specific terminologies of various engineering disciplines. The interpre-

ters of the requirements must understand the system and the terminology in the same light that the writer of the specification intends. Too often people of different backgrounds misinterpret the requirements without recognizing that misinterpretation. This is attributed (by some) to a lack of basic definitions; however, it is more likely a lack of background by the participants in each other's disciplines, and a lack of clear, concise communication on the person-to-person level.

Derived software requirements and design cannot be a process based only on paper. A classic example of this occurred on the Safeguard Ballistic Missile Defense Program. The software subcontractor did not have the years of experience (in the system and its components) that the prime contractor had. After many attempts to prepare specifications that the subcontractor could correctly interpret, problems in understanding persisted (even though the two companies were only seven miles apart). The communication problem was eventually solved by moving a substantial number (about 100) of the prime contractor's systems engineers into the subcontractor's facility so that the requirements could be interpreted on a daily and case-by-case basis.

Allocation of Requirements - According to systems development methodology, requirements for embedded computer systems (ECS) are derived from higher-level weapon system requirements plus imposed standards. These requirements are manifested in specifications which, when implemented, are intended to satisfy the user's stated needs. As the requirements flow down to subsystems through the systems engineering process, they are more quantitatively detailed in an effort to distribute the design and development activity among different departments of the same company, or among different subcontractors.

Satisfaction of quantitative requirements at some level in the "flow down" usually becomes a contractual obligation on a contractor. Changes in the requirements at the contractual level require formal technical and contractual paperwork. The process of requirements allocation and modification is one of the most important functions of systems engineering.

Some contractors are not conversant with the systems engineering process as defined by Service standards. This leads to a tendency to put inappropriate levels of detail in the requirements documents, dictate inappropriate

hardware, and require use of inappropriate methods. It also constrains the design studies and makes the acquisition agent potentially responsible for the design flaws. An example of the misunderstanding is provided by a recent ground control system where the contractor was required to re-edit all of his software requirements specifications to eliminate all design information inappropriate to that level of specification.

Implications of Requirements - Implications of systems requirements on embedded computer systems ECS (and particularly software) are not well understood. Generally, many of today's weapon systems engineers do not have adequate appreciation of how much effect a particular requirement has on the ECS. The modification of a few minor requirements can sometimes have a significant effect on the size of the ECS effort. The inability to estimate accurately the size and complexity of the software effort for new technology systems remains a problem. The ability to perform good trade studies is not universally available.

For system types and technologies new to the development team, there is limited historical data on which to assess the completeness, attainability, cost and risk associated with implementing the requirements. In particular, it is very difficult to estimate the effort associated with ECS design, code, checkout, integration and test for such systems because there is no similar prior development to use as a guide. (The converse to this is also true. The F-16 ECS development went reasonably well because of the contractor's previous experience with the F-111 aircraft. Combat Grande software is also an excellent example of success because it did not press the state-of-the-art and was similar to a previous system developed by that contractor team.)

The modification of a few minor requirements can have a substantial effect on the ECS effort; and the addition of relatively small amounts of software can have a large effect on system success. An excellent example of this is found in NASA's Viking program, which was the mission to Mars. Originally, the ground support software was to receive and process, simultaneously, the data from both Mars orbiters. Since each orbiter had onboard data storage, the contractor recommended during development that alternating (rather than simultaneous) transmission and

processing be permitted. This change saved an estimated 30% of the already high projected costs of software without degrading the mission. Another issue on Viking was the ability to uplink new computer programs to the space vehicles. This requirement was added at relatively low cost and proved critical to achieving the launch date and extending the mission life of the spacecraft.

Another example, not so fortunate, deals with an Air Force weapon system that had an appropriate margin requirement for computer sizing and timing. A high-level management review group, seeking to reduce program cost, forced a reduction of the margin to save on computer hardware costs. Not only was the software more expensive to implement, but it grew in size to where the additional hardware was put back into the system, thus raising the cost and extending the schedule. There was no appreciation of the impact of the requirement by the high-level review group even though the acquisition agent predicted the eventual impact to them.

Changes to Requirements - There are strong factors that drive system changes even when the requirements are clearly understood. Threat changes, funding changes, and technological insights are three important factors. Air Force acquisition agents tend to resist frivolous changes because of the cost of modification to documentation, existing hardware, existing software, and schedule. Change activity is usually sole source (because competition is usually not feasible) and, hence, relatively expensive. Since the embedded computer often implements the system and subsystem requirements, embedded computer software is usually severely impacted by requirements changes.

Rapidly advancing technology also impacts system requirements and change. There is often a tendency either to try to achieve the latest technology (which adds program risk and often manifests itself in the necessity to compensate in software what could not be achieved in hardware); or to use mature technology without providing for pre-planned product improvement, thus making future upgrading to new technology more expensive. There is a lack of planning to consider change and to try to incorporate it economically.

Explicit Support Requirements - Many system acquisitions have an implicit requirement for supporting ECS software when the system is operational. A number of problems often interfere with the development of good, explicit requirements regarding support. The most important omission is not developing an operational concept (and, hence, a support concept) and the subsequent decision of who will be the support agent, until late in the process. Without these issues resolved, there are problems in developing cost-effective, verifiable, support requirements for ECS contracts.

There is a lack of good engineering data on which to base trade studies for support requirements. Specifically, what tools, processors, system designs, architectual designs, and software designs are cost-effective are often merely the subject of opinion. The support objectives must be a part of the system trade studies if there are truly to be life cycle efficiencies. Changing technology, standardization efforts, testing requirements, mandated schedules and budgets, software tools, and verification and validation, all impact the support equation. The acquisition agent can only respond in his decision making to qualitative factors since he has no quantitative data on alternatives.

Concurrency of Air Crew Training Devices (ATD) - Concurrency is having the configuration of the trainer always correspond to the configuration of the primary weapon system, as far as training functions are concerned.

The configurations of the primary weapon system and the air crew training device (ATD) must be closely synchronized to provide valid training on the ATD. The alternative is to generate misleading learned responses, referred to as "negative training". For example, if a new avionics capability is added to an aircraft OFP, but not incorporated in the ATD model of the OFP, the crew members being trained will practice and learn the outdated responses in the ATD, then have to relearn the affected operations and tasks while flying. To avoid this situation, there is a support requirement for the timely updating of the ATD to reflect changes to the configuration of the primary weapon system so that aircrew members can become familiar with the modifications before trying to fly or operate the actual system.

In Air Force direction AFR 57-4 it states, "Trainer modifications should lead the weapon system modification by at least 90 days..." without delaying the weapon system modification release. In a support environment, this involves extensive coordination by the supporters of the weapon system and the ATD. Although AFR 57-4 advocates a 90-day lead time, this is seldom achieved because of funding delays and administrative restrictions. In addition, the ATD is slaved to the weapon system and the ATD support personnel are dependent on their primary weapon system counterparts for all technical information.

Concurrency of ATD change to primary weapon system change normally does not exist. Delays of ATD configuration updates of several years are common. Contracts for ATD modification are normally not let until all primary weapon system modification data is received, which causes a delay of at least 18 to 24 months for the approximately 90 percent of concurrency modifications which have to be contracted. Sometimes the delay is longer than two years due to lack of immediately available funding.

In summary, there is a long lead time from the fielding of a change to the primary weapons system (e.g., the aircraft) until the modification of the trainer. The situation is contrary to directives which call for concurrent or leading modification.

Lack of concurrency undermines the usefulness of the trainers. Pilots/crew members lose confidence in the devices, do not reduce their flying hours, and training objectives are not met. Any improvement of the support posture should be aimed at reducing the time lag to modification of the trainer.

Some Requirements for Security - The problem of ensuring the security of software, and the operational systems they support, requires the development of secure operating systems. In military systems there is also a requirement for protection of classified information and privacy data.

The design of secure software requires a balance between requirements, policy and operational requirements and technology. Additional costs of ten to fifteen percent are typical.

Early attempts to achieve secure software by use of patches did not work. The security features must be included in the basic operating system design. The use of a secure kernal has been successful in early developments but work is already underway on the development of complete secure operating systems such as Provable Secure Operating System (PSOS).

Secure software systems must be certified to verify the multi-level system security properties and to prove the design is secure. The goal of a security system is to provide a strong assurance that it is impossible for an unpriveleged user to compromise protected information.

Security considerations, in the view of inter-operability, cause increased complexity due to the number of systems and levels of security required within each system. There is need to provide a common security module that will provide adequate multi-level security for Joint Services and NATO inter-operability in a hostile environment.

Inter-operability - The technology explosion of the 1960's and 70's produced a rapidly expanding potential for the application of automation (computer technology) on the battlefield. Within every functional area, planners generated unique concepts and requirements for automated systems. Many of these systems were subsequently developed without a clear awareness of other systems under consideration, or the information exchange requirements which might result. One result of those independent development efforts is the proliferation of non-standard systems for which inter-operability requirements are not considered. Many of the systems are so far along in the development cycle that costly changes will be needed to satisfy requirements for inter-operability. But even the investment of millions of dollars will not ensure the desired inter-operability, unless: (1) interface requirements can be clearly stated; (2) inter-operability concepts are well understood; and (3) interface design planning is conducted on the basis of a totally integrated structure.

Current attempts to take this proliferation of systems, each with a unique design, and interconnect them in a manner which will satisfy a wide variety of interoperability requirements, must address some highly complex problems in network design. The following paragraphs highlight a few of the problems which must be addressed.

Intra-System Inter-operability - U.S. Army Tactical Automated Systems (TAS) are composites of several individual pieces of equipment, each of which could be considered a complete system in itself. The integration of these inter-operating subsystems has forced the TAS developers to address inter-operability concepts within their individual TAS design. Rules and conventions had to be established to govern the information exchange between the subsystems. Such intra-system inter-operability has resulted in de facto protocols which are unique to each particular TAS.

Intra-System Communications - The existence within some TAS of inter-operating subsystems which must be geographically separated, has created a requirement for intra-system communications. Since the available means of communications, as well as the information exchange requirements, may differ for each subsystem interface, different communications protocols (as well as network link, and physical protocols) are sometimes implemented in the architecture of a single TAS.

Evolving Communications Systems - Within the military communications community there exists an excellent understanding and capability in the areas of communications network control, circuit, message, and packet switching, and general communications inter-operability. Complete digital communications systems, and automated information distribution systems are now evolving. Some of these communications systems can be considered as automated systems in their own right. However, such systems did not exist, and their protocols and characteristics were not established when today's systems were in the concept formulation and early development stages. Additionally, such advanced communications capabilities are not generally available for use by systems which are now operational, or which will be fielded in the near future. As a result, the development of some systems has included the development of perfunctory communications networking and switching systems which are embedded in the system processors. In many cases, the communications protocols are included in, and not easily separated from, the user or system protocols. Accordingly, unique communications protocols often differ from, and conflict with, the protocols of the evolving communications systems.

Unique Solutions to Common Problems - Many military terms, functions, and applications are common to the entire military community, or at least common throughout large segments of that community. However, the independent development of each automated system has produced numerous instances where identical problems have been solved differently. In some cases, there may be valid justification for a unique solution. In many other cases a common solution would be equally acceptable, but the impact of changing after the fact is prohibitive. As a result, system inter-operability is restricted because similar or related systems have implemented unique solutions to common problems.

Forward and Backward Inter-operability - The life-cycles of automated systems are varied and uncertain. Some systems may remain in the active military inventory for a number of years; other systems may be short-lived and replaced by second or third generation systems; and still other systems may disappear as doctrine and technology changes produce a completely different system to perform those functions. Development cycles and the dates on which new systems will be introduced, also reflect great variety and uncertainty. As a result, achieving inter-operability today means that: backward inter-operability is required with systems, protocols, and techniques which were conceived and developed 20 to 30 years ago, and forward compatibility is required with concepts which may not be realized until 20 to 30 years in the future.

Monolithic Protocols - Many of the emerging automated systems have implemented monolithic protocols incorporating all functional levels into a single-level "ad hoc" protocol. It is often difficult or impossible to differentiate and extract particular functions from these systems. Inter-operability planners will find it is difficult to incorporate monolithic structures into an inter-operable architecture which is based on layered or multi-level protocols.

Key software inter-operability consideration is involved in the following: man/machine interfaces, software versus firmware, flexible message generation capabilities, software inter-operability training, provision of adequate multi-level security for joint Army/NATO inter-operability and considerations for continuity of operations, and survivability of automated systems in the battlefield.

PROBLEM: A.2 Life Cycle: Management

DESCRIPTION:

Knowledge of ECS - Managers who have little understanding of embedded computer software are thrust into positions of responsibility where their decisions directly affect the software development and support process. They often do not appreciate the increased complexity implied by the decisions they make, and, in particular, do not appreciate the system aspects of software development and support. Some managers ignore the software development aspects. Others believe that software is extremely flexible and, therefore, can compensate for shortcomings in other areas without serious impact. In conducting this investigation, the management issue is one of the two most cited problems (along with availability of people). Some of the statements often heard are:

- "Management does not understand software problems and puts them on the back burner."
- "Management does not understand software quality assurance."
- "Management is afraid of software-related decisions and fails to act."

Acquisition - In acquisitions (particularly on less than major programs), relatively junior officers with little acquisition experience and little ECS experience are often responsible for decisions involving millions of dollars. Use of fixed price contracts on high risk developments has, at times, been pursued when cost contracts would have been more suitable. Typically schedules and budgets are driven by what is available rather than by what is necessary to do a satisfactory development. Acquisition methodology is not always understood nor appropriately tailored to the nature of the particular acquisition.

System Management - A frequent management error for ECS is the belief that software and hardware can be developed and supported by different companies or agencies without a strong systems engineering authority to resolve con-

flicts and to perform the systems integration function. The Safeguard example, cited previously in A.1, demonstrates the necessity for close communication between system, hardware, and software engineering. One particular C^3 system, started in 1973, attempted acquisition with separate hardware and software contractors. The software development cost went from $10M to $40M due to requirements changes and configuration differences. No contractor was responsible for overall system delivery. The program was cancelled in 1978 because the software and the hardware would not work together.

Another related example deals with an aircraft system in which management bought the automatic test equipment (ATE) from one contractor and the aircraft components from another. Management failed to realize that internal changes to aircraft components, that may be form, fit, and function compatible, can effect the fault detection and fault isolation software in the automatic test system. No participation of the ATE contractor on the aircraft configuration control board was planned; hence, when the ATE was completed it could only fault isolate preproduction components but not production components. New software had to be developed from scratch.

Risk reduction suggests the use of prototyping of software, but management usually uses the prototype for subsequent source selection. Hence, the prototype is used "to sell" the system but does not achieve the goal of risk reduction.

There is still a lack (particularly in smaller companies) of appreciation by management for the necessity of discipline in ECS development, and the necessity for good communication among systems engineers, specialty engineers, and software engineers. Some managers have had inadequate exposure to computer and software concepts and terminology; thus, fearing the technology, they give software inappropriate attention and make either poor or no decisions. Management skills within the same company are highly varied from manager to manager in regard to ECS. Those elements of system management which the manager best understands tend to occupy most of his time rather than a balanced application of attention across all system problems.

Foreign Military Sales - The development of ECS-based weapon systems creates unique problems in the area of foreign military sales (FMS). These are due primarily to: (1) the intimate relationship between ECS software and classified foreign intelligence data; (2) sensitivity and vulnerability of ECS-based weapon systems as a result of software algorithms and data; and (3) the inability to accurately define technology transfer issues associated with ECS software. The problem is that State Department (and other personnel familiar with the international political climate) may not be aware of technical constraints, while design engineers and contracting personnel may not understand the political considerations.

These problem areas are further accentuated by associated problems created by: (1) national disclosure guidelines which do not provide adequate guidance on the issues of technology transfer and disclosure of embedded classified intelligence data within many ECS software; (2) lack of understanding within management of the issues; (3) State Department commitments for the sale and support of ECS-based weapon systems without adequate consultation with the DoD; and (4) the split responsibilities for release authority between DoD and DCI on ECS-based weapon systems. The above problems eventually show up as vague and incomplete disclosure letters. These, in turn, leave the service agency in the awkward position of trying to satisfy both State Department and DoD guidance on the sale of the ECS-based weapon system.

Concerns are surfacing regarding electronic warfare integrated programming, EWIR data release for foreign military sales, the national disclosure policy, and the intelligence that could be obtained through reverse engineering of software programs embedded in radar warning receivers and ECM pods. Recently the national disclosure policy group at OSD was briefed on the problems with EW systems and fire control radar. They agreed to discuss the possibility of developing new release guidelines and to include NASA on the OSD/NDP committee.

In one system, FMS programs were started with delivery dates in advance of normal lead times. Detailed customer reviews were required prior to issuance of the letter of offer and acceptance (LOA) to begin preliminary contractual activities essential for delivery. The preliminary dis-

closure guidelines were too vague (thus subject to various interpretations) and incompatible with the level of discussions required for contractual progress. As a result, the customer was sometimes given inaccurate impressions of the equipment he would be getting.

One foreign government was offered a standard Electronic Warfare suite. They were told at first that their equipment would be tailored for their use based on threat data provided by them and deemed releasable by the U.S. Government. Subsequent to tendering of the LOA, the delegation of disclosure letter (DDL) was issued and proved to be more restrictive than the preliminary guidelines. Resolution delayed contractual activities which eliminated any management flexibility and potentially may result in delivery of the hardware to the country prior to software. Reiterations of precontractual activities may be required in those instances where the DDL is more restrictive than the preliminary guidelines. The type of acquisition, such as production under license or co-production, will also affect data disclosure guidance.

The effect of FMS on the ECS world is also seen in the shortage of skilled support personnel. It is difficult enough to plan for and obtain the necessary skilled staff needed to support U.S. weapon systems which have an acquisition cycle measured in years. It is more difficult to react in months when an FMS case is being negotiated and signed. The short lead time results in an adverse impact on U.S. support since transferring trained U.S. support people to FMS support positions is often the only alternative.

The Electronic Warfare Support Directorate at Warner-Robins Air Logistics Command is an excellent example of the personnel problem. They (WR-ALC/MMR) are only 60% staffed against their engineering slots. Since November, 1981, there has been a 30% increase in signed EW cases, each of which authorizes an additional engineering slot. This number is expected to grow since there is a 60% increase in cases being negotiated. Though this activity is strongly driven by the policies of the present U.S. administration, signed cases result in the requirement for long term support to some degree independent of administration policies. Staffing the FMS work is difficult at best and tends to drain skilled people from U.S. ECS work.

Authority - Clear assignment of responsibility and authority for individuals and organizations is often missing. The lack of disciplined methods is partly attributable to management's lack of confidence in a highly technical process, and partly to its reluctance to constrain software professionals.

Schedule and Budget - The lack of useful data on project status makes management of software systems difficult. The difficulty of measuring progress in software is one factor, and the lack of historic metrics and standard measurement models is another.

Software checkpoints are often aligned with events external to the software schedule, such as hardware availability, political considerations, or manpower. This alignment is usually done without consideration of the realities of the software development process. Such schedules have no creditability.

Estimation of Schedule and Budget - There is a severe lack of useful metrics on software--how to gauge development size, complexity, schedule, and budget. There are no generally accepted models and the lack of good models makes estimation of schedule and cost difficult. Cost and schedule estimates are often based on unfounded assumptions, making allocation and trade-off analysis difficult. Vague requirements, complex interfaces, and continuing changes further compound the problem.

Life Cycle Planning - In many projects there is a lack of an effective plan for the management of the software over its entire life cycle. There is little available in the way of tools for tracking progress over a life cycle. Most of the project management plans and resource tracking mechanisms are internal to the contractor. The lack of automated project management tools causes improper assessments of potential increases in schedule and costs.

The lack of a good system plan causes many problems: coordination of different activities will break down, emphasis may be placed on the wrong areas, subtasks may be poorly synchronized, and changes may be poorly controlled. Good planning is difficult under the best of circumstances because of the constant changes required, but the lack of planning causes many procedural and coordination problems.

Tracking Tools - Few tools exist for tracking the changes to a software system over its life cycle. Some contractors have good tools but they are applied only during the development period; also, these tools are generally proprietary. The lack of good automated tools hampers management planning.

Evaluation Tools - The lack of performance monitors and modelling tools does not permit analysis of the efficiency of the current system architecture, therefore, any significant change in the system cannot be properly evaluated before it is accepted or rejected. In many cases, techniques and tools for the evaluation of system performance are not employed.

Disciplined Environment - Management must support and implement a strong discipline for systems and software engineering, both for acquisition and support, and for government and contractor professionals. Thorough plans must include: the use of a realistic software development plan; the application of clear, measurable checkpoints; the careful assessment of quality; and the planning for integration and testing of software and the system. One or more of these key points is generally overlooked.

PROBLEM: A.3 Life Cycle: Acquisition

DESCRIPTION:

Software Contracts - Software is normally acquired as part of a system. System procurement has historically been the design and purchase of hardware, so the rules for acquiring software are based on the rules for acquiring hardware. This causes many problems and inconsistencies because software does not have the same characteristics as hardware. Typical problems:

- The contract type is inappropriate to the situation, for example, using a fixed-price contract for a poorly-defined, high-risk software function.
- Requirements continue to change and are either uncontrolled or ineffectively controlled.
- The system contract does not specify that it includes all rights to the software.
- The software tools used to develop and maintain the software are not acquired.
- Money is not provided for development of software tools and tool documentation.
- Life cycle costs are ignored.

Management of System Interfaces - Frequently, different parts of the system are developed by different contractors. The focus is on the hardware and little attention is paid to the software. Good interface specifications and management are necessary to ensure a smooth integration of all parts. Many problems occur at the interfaces, especially changes which are known only to one side of the interface.

Project Status Reports - Documentation for project status is often lengthy but not very useful. Software does not have highly visible points at which status can be carefully tracked and measured. The concept of "inchstones" instead of "milestones" increases the number of software measurements but requires much more management and contol.

Documentation Requirements - Documentation is frequently the orphan of software development. Its requirements are vague and overlapping, it frequently gets starved for funds when hard choices must be made, it is often overwhelming in volume and marginal in value, and it is mostly written after the fact.

Change Control - Requirements changes and design changes cause many problems in software. Software is the part of a software/hardware system which is most amenable to change, so changes tend to impact software and be clustered at the end of the schedule when there is the least time to handle them. Change is endemic to software and must be very carefully and rigidly managed; it frequently isn't, with disasterous results.

Reliance on Contractors - DoD relies heavily on contractors for creating and managing the software development process. Such heavy reliance naturally causes problems. As the percentage of work done by contractors increases, the problems for DoD also increase.

Design for the Complete Life Cycle - Design decisions for software are rarely made with the complete life cycle in mind. Decisions are normally made to resolve an immediate design problem and rarely consider how the change might impact the cost of the software over its useful life. Good design can reduce the costs incurred during the long period of in-service support; this long period of maintenance is a reality for most weapon systems and is rarely considered. All resources (i.e., hardware, software, documentation, tools, personnel qualifications, etc.) needed for life cycle support should be clearly identified during acquisition, but rarely are.

PROBLEM: A.4 Life Cycle: Product Assurance

DISCUSSION:

Introduction - Software in embedded computer systems is developed for requirements that characterize the needed functions and performance of the weapon system. Implicit in the development are characterizations of the hardware (computer and otherwise), the operator/maintenance, and the physical and threat environment. During the system development, these characterizations become more realistic, i.e., hardware prototypes are more realistic than paper designs, actual feedback is more realistic than anticipated reaction, and operational test is more realistic than simulations. As more realistic behavior occurs, the software must often be modified to accommodate a new and better understanding of the system. For new systems for which there is no history of engineering data, the discrepancies between anticipated factors and real factors are magnified.

Testing - The purpose for testing software during development is to demonstrate that the product satisfies the design and the requirements. Testing reveals (although not completely) coding errors, design errors, and requirements incompatibility. Static tests focus on syntax errors while dynamic tests focus on all types of errors.

The best testing process, according to modern perspectives, is incremental during the development. That is, units are tested before integration of all of the software so that when the complete software package is tested, many of the coding errors should have been eliminated.

For extremely complex software one hundred percent assurance is impossible based on limited development and acceptance tests. Complete assurance only comes through extended use.

For different application types, different test requirements should exist since the implications and impacts of errors are different. For example, an error in nuclear-release software is far more important than an error in support software.

There is a lack of criteria for various applications to determine how much testing is best at each step in the software development. There were so many hard-to-find errors (due to insufficient module testing) during software module integration of a recent digital flight that the integration was curtailed and more extensive module testing resumed. This caused a substantial slip in schedule. Furthermore, test environments are often different from the real environment -- a factor that influences the amount of testing performed and the reliability of the software. Too often, testing is governed by the amount of time and money that are available. The area of optimal test requirements reqarding ECS software needs far better definition.

Application Dependencies - As indicated previously, the degree of product assurance depends on the severity of the implication of an error. General test tools may be helpful, but they will test only against paper requirements and paper environments. The cost-effectiveness of these tools individually and in combination must be assessed for each application type.

In the flight controls application, the software may be required to accommodate multiple hardware failures. The likelihood of crash due to software may be stated in terms of no more than one crash for every 10^7 to 10^9 flight hours. Assuring these requirements are met is most difficult.

In self-learning systems or other applications of artificial intelligence where the software may be changing itself or its data base over time, there is also an important product assurance challenge.

Continuous Process - Software testing is not a single event during system development, but should be considered a continuous process throughout the development phase, and for much of the early operational phase, of the total system life cycle. Software for current and evolving military systems is increasing in complexity to such an extent that "100% testing" of every software program module and logic path before deployment is impractical. Adequate testing to assure quality in delivered military systems requires concerted cooperation and intelligent interchange of expertise among the system development agency, prime contractor, formal test agency and user. Effective and manageable test plans, test procedures,

test facilities, and simulation capabilities are as important to the assurance effort as these elements have been to hardware testing. All too often, however, test planning for software has not received an adequate share of total resources. The complexity of software design requires automated aids to validate and test software products, and significant time to verify that the product reflects its design parameters. It is all too frequent that significant shortcomings in functional and development specifications are uncovered only by errors occurring during software testing and analysis. These errors are then attributed to the "software", when they are more properly tied to incomplete or vague functional specifications.

Independent Verification and Validation (IV&V) strategies and methods have evolved to help achieve better results for assurance. Since the IV&V process is costly, continues throughout the development cycle, and is a source of critical review, it is difficult to implement on a sustained basis. IV&V is often perceived as not being vital to system development, and unnecessarily duplicative of traditional design testing and assurance functions. But IV&V, properly implemented early enough in the system life cycle, can avoid surprises during development due to design flaws or requirements inconsistencies.

As automated systems become more complex, the software which runs those systems also increases in complexity. Simple methods which were adequate in the past for testing and certifying operational and applications software, will not suffice in today's environment. Sophisticated software tools, test drivers and automated program test aids are needed to provide assurance on a timely basis during development testing and operational testing phases. Such tools must themselves be certified and appropriate to the task. Statistical models may be applied to the testing of complex, multi-interface systems. Each overall program schedule and its application of resources must recognize the essential need for software testing, and provide it throughout the program life cycle.

PROBLEM: A.5 Life Cycle: Transition

DESCRIPTION:

Until the DoD Acquisition Improvement Program, the DoD had no guidelines on how to handle rapid technology advances. These initiatives provide some answers to the following questions: "Should the most recent technology be incorporated, or should state-of-the-art be used with a planned product improvement approach to follow? Should systems be required to be designed to incorporate new technology in the future? Just how should technology transfer take place--through specification or through contractor initiatives? Lastly, how can acquisition and support policy, procedure, and skills keep up with the technology explosion?" The new DoD directive favors the planned product improvement approach. This concept for ECS has not been clearly identified, nor have its implications been determined, nor its acceptance been assured.

Rapid Changes in Technology - The very rapid change in computer technology has a destabilizing effect on acquisition and support engineering. For example, microprocessors and firmware are widely used in today's weapon systems, yet wide acquisition familiarity, policy, and support concepts have not yet caught up to the technology. Contractual requirements intended for software had been disputed by contractors as being inapplicable to firmware, hence appropriate documentation had to be purchased as a contract add-on. Rapid evolution impacts the availability of older devices to be used as spare parts for installed older systems.

The economics of hardware and software costs are shifting. Past engineering data is becoming obsolete more rapidly. The arrival of Very High Speed Integrated Circuits (VHSIC) will cause another new technology impact that must be absorbed even before the microprocessor/ firmware issues may be fully resolved.

Microprocessors and Firmware - In current systems, microprocessors and firmware are often used as cost-effective replacements for previously-used special purpose analog circuitry which was embedded in electronic components.

The use of commercially available digital components as analog replacements not only improves the accuracy of the composite device but also reduces its cost. On the other hand, the firmware resident in these devices represents computer software.

Since firmware involves software, there are major acquisition questions that must be addressed:

- What documentation should be required for acquisition visibility and support?
- What standards should be imposed on the firmware during development?
- What levels of testing should be required of the firmware during acquisition and support?
- What development tools and facilities should be acquired for subsequent firmware support?
- What architectural design requirements, if any, should be placed on microprocessor arrangement and firmware?

There is a problem relating to the development and support of microprocessors/firmware in the areas of identification, labeling, definitions, etc. The problem stems from the practice of attempting to force these devices into either software or hardware guidelines. There must be a distinction made. However, when this is done, it is usually agency-unique leading to problems in transition from agency to agency.

Formulation of a uniform policy concerning microprocessor and firmware definitions, identification/labeling, concept of operations, configuration management practices, HOL policies, and DIDs are needed to alleviate these transition problems. For example, firmware, when programmed, may have an almost unlimited number of possible configurations which are machine readable but which require specialized equipment. Combining this with the fact that both the program and the media embodying the program must be controlled, adds to configuration management implications. Labeling to identify the media, along with internal configuration of the device and/or board, further complicates matters if these techniques are not well defined and standard.

It is very clear that there is no single cost-effective answer to each of the above problems for all applications. For a deeply-embedded firmware application (where the firmware developed is integral to the hardware of the component in which the microprocessor is embedded) there is one set of answers. For the application where the computer program is expected to change frequently and firmware is used (in preference to software) for cost or other technical reasons, the answers are different. There is a highly application dependent flavor to the answers of each of the previously listed problems.

Impact of Future Technologies - The impact of future technologies on ECS must be studied and guided. Skills in new technologies are, by definition, limited. Impacts of new technology on software design methodology have yet to be determined. VHSIC and VLSI (very large scale integration) still have unknown implications on system architecture and software engineering. New advances such as enhanced modular signal processors and highly redundant systems must be quickly assessed as to what their hidden software implications may be. The rapid technology change must be a factor in the ECS standardization efforts with guidelines to assist in establishing levels of standards and conditions for changing standards. In short, our systems acquisition process, support process, and policies must have the flexibility to adapt to technological change.

Support of Automated Systems - Substantial support problems have been built into automated systems during the development process. A principal factor has been the inability of the developer to develop and validate system and tactical software requirements, evaluate doctrinal problems, and perform user acceptance testing on a systematic basis. Poorly designed requirements result in costly delays and redesigns of systems and their associated software, and have caused extensive modifications after fielding. The problem is compounded by the sensitive nature of the evolving systems software. Therefore, the developer must be provided adequate facilities and other resources to perform his mission and to provide adequate guidance to field users on employment of these automated systems.

Problems During Development Process - Achieving effective post-deployment support for a system is greatly influenced by actions taken during the development phase. Specifically, consideration for support must be planned for and built in during the development phase. In this regard, there are a number of current deficiencies:

1. In general, the present management structure at all levels has not been ready (or has lacked the resources) to implement the changes necessary to develop and support automation. Specifically, a structure for research, acquisition, development, and post-deployment support is needed. Early and continuous involvement of independent testing and support organizations with the developer is another recommendation.

2. No integrated set of procedures, guidelines, and standards are uniformly applied to the development of automated systems.

3. One problem is the lack of identification, and disciplined use of, an integrated, effective procedure for system development. A plethora of regulations, instructions, and directives are currently used by the Services. These documents address the development of specifications and documentation as well as procedures and methodologies for quality assurance, configuration management, and software development.

4. In general, a lack of concern during software development has produced software with poor attributes for in-service support. The software in automated systems is characterized by poorly defined function and interface requirements and specifications, lack of proper modularization, too great a use of machine-oriented languages, and inadequate documentation.

Coordination of R&D - There is a need to coordinate the full spectrum of R&D activities (Categories 6.1 through 6.3) within the R&D community. This coordination would smooth the movement of R&D work to operational use in fielded systems.

B. Environment

The term "environment" includes all of the activities involved in the development and in-service support of embedded computer software. Problems included as part of environment were divided into the four major sub-categories of disciplined methods, tools, reinvention, and capital investment. The problems described under environment are more related to "how" software is developed than to "what" is developed. The problems are interrelated within the environment category as with the other software problem categories.

PROBLEM: B.1 Environment: Disciplined Methods

DESCRIPTION:

There is a wide range of system engineering and software engineering practices within DoD and its contractors. The practices range from well-disciplined approaches for creating high quality software to undisciplined approaches whose product cannot be controlled. DoD needs a consistent approach that ensures quality software.

The current state of software engineering practices is a hodge-podge of tools, techniques, and methods, and exhibits a number of serious weaknesses. The variation is both good and bad. The many ideas being tried reflect the current nature of software technology. The variation also inhibits movement of people because of retraining and adds to managements problems in trying to do the work.

Contractors have a strong interest in trying to reduce costs. They invest in software tools and software education to promote productivity. However, these tools rarely leave the contractors shops. Contractors tend to have a stable set of tools which evolve over time. Government shops are also stable, but have to cope with the delivered software, no matter what tools are, or are not included with it, and have to support many types of software over a long period.

The lack of consistent, disciplined methods impact embedded computer software because the software is developed by many independent groups. The lack of consistency and undisciplined methods make supporting such software difficult and costly. As a result, software is difficult to manage, varies widely in cost, quality and maintainability, and makes it extremely difficult to improve in the sense of technology or quality.

The lack of a well-developed system or software engineering process is evident everywhere in the embedded software. A well-defined software engineering process involves a set of activities for developing and supporting software through its life cycle. This process becomes the framework for a set of consistent, integrated methodologies. A well-conceived methodology is a repeatable human procedure which separates the clerical aspects of an activity from the creative, intellectual aspects.

Today's software includes elements of the above but not in a coherent form which integrates separate activities into a consistent whole.

PROBLEM: B.2 Environment: Tools

DESCRIPTION:

Software is a labor-intensive technology. The EIA report[1] predicts that the software and services portion of DoD computer system costs will increase from $4.6 billion (or 69%) in 1980 to $37.2 billion (or 81% of the total) in 1990 (figures are 1980 current dollars). Note that software cost already consume 69% (as of 1980) of the DoD budget for computer systems.

The inefficient use of people is one problem frequently cited for this task force. Software involves both creative and mechanical activities. Most of the investment in software tools and technology has been to reduce the time spent on mechanical activities, such as entering data, very little investment has been made to enhance creative activities. Creative work involves manual recording as ideas are developed, and extended or discarded. Thus, a significant part of software development is the process of manually recording information. Automation of these mechanical processes is occurring (e.g., interactive terminals) but more is needed to improve productivity and to establish more consistency and discipline.

To increase productivity in ECS design, coding, test, and support, as well as development management, there has been emphasis on tools and procedures. To avoid a problem of proliferation in the support activity, the Air Force has considered common and modular tools that could have wide-spread application. This introduces a number of new problems:

- First, there are no widely accepted productivity measures for various tools nor combinations of tools. Without such measures, and corresponding productivity data, one cannot conduct trade studies or assess whether or not the use of the tool is warranted.

---

[1] "DoD Digital Data Processing Study -- A Ten Year Forecast," presented at the Electronic Industry Association Fall Symposium in October, 1980.

- Secondly, for government developed standard tools there are other problems that must be resolved. First, the availability of the tool may constrain a contractor to a particular and unfamiliar, support computer. This, in turn, may prohibit the use of his own tools with which his people are familiar. Also, for government furnished tools, which are required to be used, the tool and its documentation must be nearly faultless (i.e., meet specified requirements such as correctness, performance, optimization, accuracy, and robustness); otherwise, the government is liable for contractor misuse of the tool or resulting system and schedule problems associated with the faults in the tool. Lastly, standard tools and environments tend to stagnate innovation in ECS development.

With these cautions in mind, uniform, automated software design and support tools are needed which are portable, where necessary, yet tailored to different ECS applications and phases of the life cycle. If a development approach using modules was used, the building blocks or components would become technically stable, thus combining into more reliable, quality systems.

In addition, as productivity trade studies and new technologies lead to other tools, modular enhancements can be made to an existing integrated development and support environment.

Some candidates for automated tools with modularity are:

- Software documentation system
- Configuration management system
- Data base management system
- Management information system
- Software libraries
- Improved ECS analysis tools (simulations, diagnostic emulations)

- Comprehensive software development, support and test tools (specification development system, editors, code syntax checkers, scenario generators, ISF and operational test data reduction software, etc.)

Industry has been working on many of these tools. Their effectiveness on productivity, both individually and in combination, must be assessed for various application types.

Achieving Tool Uniformity - The approach used in developing the F-15 software support facility illustrtates the level of success possible in achieving tool uniformity between today's fighter aircraft. The F-15 Avionics Integration Support Facility (AISF) was developed internally by WR-ALC/MMEC, beginning in late 1976. The approach taken for development was to be as common as possible with the existing F-111 facility. This approach was intended to serve two purposes: software would be transportable between facilities and, by keeping the design concept as similar as possible, a model would be available for scoping, pricing, sizing, etc.

The design approach involved a common support concept, common processors and peripherals, common language (where possible), and other common hardware where feasible. Also, contributing to the decision to adopt a tri-Harris architecture for driving the dynamic simulation, was the fact that the F-15 trainers had a tri-Harris/4 processor and the OFP development software resided on a Harris/4 configuration. While it was apparent from the beginning that a majority of the F-111 application software would not be directly applicable to the F-15 due to the vast difference in weapon system mission and avionics system architecture, it was believed that much of the top-level design and many algorithms would be reuseable.

This reusability turned out to be true. While only a minimal amount of software was directly reused, much of the previously developed F-111 and F-15 software was used by extracting approaches, algorithms, and some code. It was the opinion of personnel closely associated with the F-15 development that first, the development of the F-15 facility internally would probably not have been attempted without the F-111 model to follow, and secondly, the model was of immeasurable benefit in establishing requirements of the facility. It must be noted that the F-15 AISF proceeded through development without major modification

of requirements. It was also the universal opinion of personnel closely associated with the background of the F-15 facility planning, that facility requirements had been grossly underestimated prior to using the F-111 facility as a model. A composite of the commonality between the two facilities is shown in Table A-1.

As can be seen from this summation, virtually all aspects of the facility was maintained as common as possible. While the benefits from this approach are not quantifiable, it is highly probable that inadequate support would have occurred if this approach not been taken.

Considerable effort in standardization of higher order programming Languages (HOL) for ECS applications has been expended. It is still necessary to establish more clearly the productivity improvements in development and support for various application types as a function of requirements for efficiency, safety, rapid modification, and correctness.

Support Facilities -- Many facilities for inservice support of software are hosted on computers designed for embedded systems. Some of these computers are seriously constrained in size (memory, speed and instruction set architecture), support hardware (on-line interactive terminals, disk space, etc.) and support facilities (utilities, data management systems, debuggers, test tools, etc.).

	F-15 AISF	F-111 AISF	COMMENT
Dynamic Simulation Processors	Tri-Harris 7	Tri-Harris 4	Software compatible, also compatible with F-15 Trainer
Host Processor System	Interdata 8/32	Interdata 8/32	Identical
Graphics Hardware	Adage	Adage	Same-differed in software implementation
Terminals	HP 2645A	HP 2645A	Identical
Peripherals	-	-	Virtually identical, compatible
Dynamic Simulation Shared Memory	Yes	Yes	Compatible
Data Reduction and Analysis System Hardware	EMR 7000	EMR 7000	Identical
Dynamic Simulation Operating System	VULCAN	VULCAN	Identical
Host Processor Operating System	Mod Interdata	Mod Interdata	Identical
Cockpit Mockup	Lab set-up	Lab set-up	Similar
Simulation	-	-	Used similar approach and many algorithms
Overall design Approach	-	-	Similar
Dynamic Simulation Processor Tasking	-	-	Same

Table A-1: F-15/F-111 Common Facilities

Machine and Language Proliferation - Currently, more than 44 different computer languages are in use on Army battlefield automated systems. Types of language range from high-order languages (HOL) to assembly and machine languages and microprocessor instructions. A survey indicates there are 10 HOLs in use. These HOLs are COBOL, EUCLID, FORTRAN, JOVIAL, PASCAL, PL/M, TACPOL, CMS-2, ATLAS-E, and MOTEL.

The situation is exacerbated by the number of dialects that exist for a given HOL. For example, FORTRAN programs are not necessarily transportable across machines because of deviations in language definition and implementation.

The survey also shows at least 24 different assembly languages, and at least 10 different microprocessors. Clearly, the proliferation of languages increases all types of support required from training to documentation. This proliferation is not technically necessary but results from the autonomous development of each system.

DoD instruction 5000.31, "Interim List of DoD-Approved High Order Programming Languages," provides a first step to curtail the proliferation of languages within the DoD by requiring a system developer to choose from a list with a limited number of HOLs when a new system is developed. An effective interim language standardization program in the Army, and use of the DoD standard language, Ada, when available, are essential to the Army to minimize support costs and maximize the availability of programers trained in the approved language.

Lack of Consistent Standards for Software - The lack of consistent standards causes many problems in the software acquisition and development process. The multiplicity of conflicting standards, and the problems in interpreting these standards forces software professionals to spend time studying variations of standards instead of producing. Good standards are missing in the critical area of software quality.

Application of Consistent Standards for Software - There is a great variation in the software development practices and standards employed across DoD projects. The lack of uniform standards can contribute to maintenance difficulty, and re-engineering of the software upon transition to a new support group.

Lack of a Standard Tool Set - Many tools are needed for developing software. These include: editors, debuggers, library managers, data base managers, high level languages, etc. These tools are often not available, unpublicized, difficult to understand or use, or inefficient. Software professionals are often unaware of the tools and models that could save them considerable amounts of time.

Inadequate Support Systems - The computer support of software development is inconsistent. A real need exists for a system that supports the full life cycle of software development. The system should be fully automated and support the full cycle: requirements analysis, specification, design, implementation inservice support (for enhancement and correction of errors), correctness analysis and management.

Promulgation of Tools - There are difficulties in transitioning generic or non-weapon specific software tools (general tools developed by laboratories) from 6.3 programs to weapon systems programs. The reasons for this are perceived to be the following:

- The risks associated with GFE (performance of tool accurate documentation, government liability on adverse schedule impact).
- The need for continued maintenance of the tools by the government.
- Competitive implementations by industry.

The fielding of the Air Force JOVIAL J-73 compiler should provide some insight into this problem.

Inadequate and Non-Modular Tools - Software problems associated with military embedded computer systems are well known. "Excessively expensive, untimely, difficult to maintain, non-responsive to user requirements, inflexible to change, unreliable. . ." are all common criticisms of most systems developed to date. The inadequateness and lack of standardization of the development and support environment has cosntributed significantly to the perpetuation of many of these problems.

One aspect of the environment involves the software tools which can provide support in virtually all aspects of the programming process. However, support tools have fallen far short of their potential value. It is worthwhile to consider some of the problems which have impeded the progress of support tools to date.

- Non-Reusability. The proliferation of programming languages and computers has given rise to a situation where most developers find themselves with a unique combination of language and host/target processors. Consequently, existing tools cannot be used and new tools must be developed.

- Minimal Tools. Because tools have been developed by the project for the project, resource constraints (manpower, funding, time) have usually compromised tool development in favor of the embedded application. The result is that only the minimum number of essential tools are developed. There have been cases where a language processor (compiler/assembler) was the only tool available.

- Lower Standards. Because the embedded system is the focus of a development and not the tools, which are often not even deliverables under a contract, a significant lower level of quality results. Tool designs and implementations are typically ad hoc and poorly documented.

- "Bag of Tools" Approach. Tools have usually been viewed as semi-independent functions. Their implementations reflect that view and the result is a "non-uniform," often "unfriendly," user interface. A command to one tool is often very different than the command to another tool, even when the objective is identical. For example, END, STOP, EXIT, HALT, might appear across several tools as the command for stopping. A "uniform" interface would allow the use of a single command, independently of the tool being accessed.

  Some tools are designed in a way that makes their use extremely awkward. Command strings like **/% and the absence of feedback prompts to the user, make a tool very clumsy to use and present an unfriendly interface.

The "bag to tools" approach also causes common subfunctions across tools to be unrecognized; the result is unnecessary duplication.

- Language Independence. Well intentioned tool designers often seek to maximize the utility and flexibility of a particular tool and strive for language independence rather then support for a specific language. Their objectives have been narrowly focused on the tool rather than the larger environment of which that tool is a part.

Each of these problems has contributed significantly to the present day software dilemma of embedded computer systems. If they are to be solved in the future, a new view of the entire environment and tools in particular, must be taken.

A System View - It seems clear that if a dramatic impact on the software problems is to be achieved, support tools must be viewed as a system having five broad objectives:

- Integration. Each tool element must be considered in relation to the entire environment and not narrowly founded. Tools ought not to be viewed as a collection of independent entities, but rather an integrated set of cooperating functions which are designed to provide a uniform, powerful, and friendly interface.

- Support Throughout Entire Life Cycle. Tool use should not be limited to development activity. In service support, which represents a significant portion of life cycle cost, should be viewed as an extension of development and not as a detached exercise in field "patching" (because the percent is system dependent). A tool system should allow support to occur in the same environment as that of development. The complete history of a program should be available to implement and validate post deployment changes.

- Standardization. Tools should be standardized to the maximum extent possible, and designed and implemented for the ease with which they can be transported and reused across a number of host processors.

- Support of the DoD Common Language. The tools of the environment should be oriented toward a single higher order language and should encourage and strengthen its use. The DoD common language (presently referred to as "Ada") is an appropriate choice as the central language. Whenever the dilemma of generality verses stronger support for the DoD language arises, the latter should be chosen.

- Flexibility and Maintainability. The tool system should be designed in a way that simplifies the ease with which evolutionary changes can be incorporated. Changing programmer needs and technology advances will require an extremely flexible tool structure. The tool system should be carefully designed, well documented, and written in the DoD common language to simplify the tool maintenance problem.

An initial tool system might consist of (but not be limited to) the following major functions:

- A text editor strongly focused on the DoD common language for entering and modifying source programs. The editor would be language structure oriented (rather than line oriented) and would perform elementary lexical and syntactical checking.

- An Ada Compiler which accommodates the language requirements and whose design is modular and compatible with the requirements for secure operating systems.

- A configuration/module manager which oversees the activities of module generation, interconnection, and maintenance in an environment strongly oriented toward the language.

- A linkage editor which integrates separately compiled modules into larger units.

- A symbolic program executor (static debugger) which provides statisitical information based upon the symbolic execution of a source program.

- A collection of documentation aids which assist in the generation and modification of software documentation, and which supports the user in selective reading.

- A dynamic symbolic debugger which supports program debugging in the target environment at the DoD common language level.

PROBLEM: B.3 Environment: Reinvention

DESCRIPTION:

Many software design and implementation problems are well understood and continually recur in new systems. Unfortunately, the designs and implementations of previous systems are not captured and reused in succeeding systems. Thus, a significant opportunity for productivity improvement is lost.

Two elements are described to address this serious problem: reusable components and program skeletons. Reusable system components are commonly recurring components which can be designed and implemented once, with appropriate parameters for tailoring to different applications; these components might include firmware. Program skeletons provide the structure into which application-specific modules can be incorporated.

Three elements are needed to support and encourage reuseable components:

- Interface Standard - A rigorous interface standard must be provided for all candidate reusable components; it would govern design, implementation, and documentation.
- Library - A library of reusable components must evolve for use by all development activities.
- Index - An index of reusable components including summary descriptions and specifications, must be provided to help the designer locate appropriate components.

Program skeletons require a similar set of support elements:

- Design Standard - A rigorous design standard must be provided for any major software system or subsystem to be implemented as a skeleton.
- Skeleton Library - A sparse library of subsystem or system skeletons must be provided for use in new development activities.
- Supporting Tools - A set of supporting tools must be implemented to aid in the expansion of the skeletons.

PROBLEM: B.4 Environment: Capital Investment

DESCRIPTION:

Limited Support Facilities - Many of the DoD computer support facilities are over-loaded and have aging computer equipment. Many are even hosted on target computers - the computers which are embedded in the weapon systems. This penny-wise, pound-foolish approach maximizes the value of computer hardware at the cost of people, a much more scarce and costly resource. The support facilities need more power to reduce the necessity for professionals to do repetitive, mechanical tasks. Computers are supurb at doing detailed, repetitive work and should be used to do more of it so the professionals can concentrate on the creative, intellectual work, work which computers cannot do.

Use of Target Hardware as Host - The use of target hardware to develop software is severely counterproductive. Target hardware is designed with stringent size and weight constraints to perform a specific military mission, is it not designed to support software development. The many tools, utilities, testing aids, text editors, libraries, and data base managers needed to support development simply don't exist. Commercial computers also reduce training costs because more people know these systems. Plus many more tools exist for commercial computers. There is a one time investment necessary to replace target computers with commercial computers to host development, but the benefits are significant.

IR&D for Software Technology - During this investigation, several assertions were made that software technology suffered in the Independent Research and Development (IR&D) arena. The consensus of those discussing the problem was that software projects received lower scores than hardware projects during IR&D reviews. A number of reasons were given for this problem, including:

- IR&D evaluators are not often software oriented. Some thought that part of this reason revolved around the fact that software technologists thought of IR&D reviews as mundane work not worthy of

their effort. Others thought that DoD did not place enough emphasis on getting the right people to the reviews. It is probably a little of both.

- Software results are not tangible, that is, you cannot touch or see the results.

- There is a lack of good ideas for software research projects. This reason is the most far reaching of the three, and is the most difficult to resolve. Some took the position that "ideas attract money" rather than the reverse. Some very knowledgeable people felt that IR&D was sufficient to fund the really good software ideas.

While there was considerable agreement that software received lower scores, there was not total agreement on the effect on IR&D software expenditures. Generally, it was perceived that some companies let this affect the distribution of resources; however, some felt that software oriented companies had no choice but to place IR&D resources in software projects, and did so.

C. PRODUCT

The software product is frequently critized. This is partly because software must solve many of the problems in the hardware and these are usually discovered late in the development cycle. However, software also deserves much of the criticism directed at it. It is costly to develop and support, difficult (if not impossible) to understand completely, and its performance cannot be measured very well. The product section addresses the utility of software, metrics, design attributes, and documentation.

PROBLEM: C.1 Product: Doesn't Meet the Need

DESCRIPTION:

Ambiguous Requirements - Ambiguous, unclear, and incomplete requirements coupled with inadequate communication between the user and implementor often causes undetected omissions or internal contradictions. These eventually result in changes as clarifications are made to the system to correct the problems. Close cooperation between users, implementors and many others is vital to the development of a useful system.

Completion Criteria - Since completion and performance criteria are often inappropriate, incorrect, and impractical, product evaluation is difficult. The lack of "lessons learned" feedback to the software development community is a serious omission since such a mechanism could provide useful information. The lack of completion and performance criteria often lead to additional problems such as determining when the system "works" (i.e., meets performance specifications), or determining when the system is "complete" (i.e., meets completion criteria). Such problems are non-trivial yet are common.

Product Failures - ATE software and its ability to sufficiently test line replaceable units (LRU) and shop replaceable units (SUR) has not received a high level of attention from management in the past. However, there are disturbing trends in test results of our more complex black boxes and boards. While it is agreed that there are multiple reasons for CND results, such as personnel proficiency, intermittent failures, etc., there are strong indications that incompatibilities between test equipment at the different levels of maintenance contribute heavily to this situation. In a majority of cases, this is due to software inadequacies or can be corrected by software. The result of excessive CND rates is additional spare cost to the service agency, unavailability of prime equipment because of shortage of replacement spares, and/or additional maintenance and test costs.

An example of the severity of the problem can be seen from some Air Force data on LRU and SRU RTOK/CND rates related to a front line weapon system. Some of the more complex black boxes are experiencing RTOK/CND rates in excess of 40%. In the short period of this study, it is impossible to determine the costs associated with the part of this per-

centage that could be reduced by better software or enhanced testing methods. However, it is believed by very knowledgeable individuals that a large portion could be eliminated. The points made during this study indicate that further study is warranted.

Also, see A.1 Life Cycle: Requirements for additional details on problems that are closely related.

PROBLEM: C.2 Product: Software Metrics

DISCUSSION:

There is a lack of good analytical methods and hard empirical data for software. It is needed as estimating information for future cost and mission impacts which are associated with decisions on embedded computer systems (e.g., design methods, quality metrics, cost data, etc.). The allocation of resources and estimation of costs associated with embedded computer software development and support are directly impacted by this lack of data.

There exist many software costing models and considerable amounts of data on software development; however, the software cost models require inputs, such as size and complexity of code, which are very difficult to estimate early in a program. Reasonable estimates can only be made when the software requirements and preliminary design are complete (which is often after a contract has been signed). Existing software metric data lacks the necessary parameters to permit correlation with corresponding parameters for new systems.

Individual companies collect data and corresponding parameters on their own projects, but are reluctant to share this data with the government since the information represents part of that company's competitive edge and negotiating potential.

There are no validated models of life cycle costs and productivities in ECS development and support beyond cost estimation data. For example, difficulty with using current models can be seen in these figures taken from a guidebook on software cost estimating:

Project	Forecast Total MM	Actual Total MM	Ratio Forecast Actual
A	419.7	71.0	5.9
B	2288.5	991.7**	2.3
C	51.5	43.8	1.2
D	3298.7	514.8**	6.4
E	7.9	7.3	1.1

(** Contains some estimate-to-complete data, along with actuals.)

While such terms as supportability, correctness, useability, transportability, reliability, among others, are used to describe qualities of software, their interpretation varies. The priority which one places on a given quality factor may also vary according to system application. To be contractually useable, these quality factors must be carefully defined and capable of being measured and verified in an objective manner and must drive the acquisition to the desired result. Additionally, threshold values for quality factors are needed which will allow the acquisition agency to state their order of importance for a particular software system.

Cost, schedule, and sizing data need to be gathered from which models and metrics can be derived that can be commonly agreed to within DoD. Projections of future productivity improvements, learning curves, and skill mix must be a part of these models and metrics.

Hardware/Software/Firmware Tradeoffs - The rapid advances in computer hardware technology, specifically chip-size microprocessors, has added firmware to the lexicon of the computer field. Traditionally, we have thought of a computer resource as a marriage of hardware and software - two clearly separable configuration items. Firmware, a combination of hardware and software, is often being given equal status with software and hardware. A number of definitions of firmware are in vogue, the most common state that firmware is:

- Reprogrammable hardware
- Hardware implementation of software
- Hardwired computer software
- Microprocessor software

For our purpose, firmware is best defined as a computer program that has been physically fixed in the memory of a processor. The program, which may have been written in a higher order language (HOL), exists in binary code which is permanently stored in a memory device (i.e., read only memory or ROM).

The designer of embedded computer systems (ECS) has the option, with today's technology, to allocate [illegible] functions among the hardware, firmware, and software of a system. This design opportunity introduces a [illegible]

of concerns which must be addressed early in the system life-cycle. Achieving the most cost-effective design will require thorough trade-off analysis of the technological, acquisition, operational, and support aspects of the system.

The added dimension of firmware introduces more questions to be resolved and decisions made. Among the questions that need to be addressed are:

- What functions should be allocated to firmware?
- How will the firmware be documented?
- What management and control procedures will be applied to the firmware?
- What is the firmware maintenance philosophy?
- How will the firmware be supported?

The basic question, underlying all of the above, is whether firmware is a subset of hardware, or a subset of software, or a separate entity. The choice, which is essentially a design decision, will determine the treatment of specific firmware elements of a system. Standards, procedures, and documentation requirements need to be updated to aid in the decision process and to accomodate the selected firmware option. Some firmware, which is "hardware intensive," may be treated as a configuration item (CI) while "software intensive" firmware could be covered as a computer program configuration item (CPCI). The intermediate categories of firmware need to be evaluated as to whether they should be treated as, perhaps, a firmware configuration item (FCI) or, as has been suggested, as a computer program in hardware subset of a CI.

The problems associated with using firmware in an ECS relate directly to the likelihood that a hardwired software (i.e., the software part of the firmware) will change during the ECS life-cycle. At one extreme is firmware that is not intended to be changed. With a near zero probability of changing the firmware logic and code, this should be classified as part of a hardware CI and treated accordingly. At the other extreme is firmware that should be adaptable to changing operational requirements and thus would be expected to be easily reprogrammable. In fact,

one of the trade-off considerations would be whether this should be handled as downloadable software rather than as firmware. The close similiarity of this software-intensive firmware to software strongly supports treating it as a CPCI.

The intermediate classes of firmware have attributes of both hardware and software in varying degrees. They fit neither CI nor CPCI, but rather some combination of the two. Changes to the hardwired software are not planned although they are expected to occur. The decision-maker is faced with the same question throughout the acquisition process, should CI or CPCI standards be applied to this problem? The real problem in achieving a cost effective design in the presence of firmware is to correctly assess the changes that may be necessary during the life of the system and plan accordingly. Perhaps a firmware documentation package is needed. As a minimum, the current ECS documentation package should be reviewed and adapted for use on firmware.

Impediments to Productivity - Human productivity in the areas of software design and implementation has been studied as a metric for several years. It is difficult to isolate the creative aspects due to its basic nature and to the lack of consistent software languages, tools and methods. Implementation is difficult to isolate, but less so, as its basic nature and tools and methods are better defined. Separation and measurement of creativity and implementation may improve as Ada, with its improved metrics, becomes widely used. Once the impediments to productivity can be identified, improved methods or tools can be developed to eliminate some of the bottlenecks.

PROBLEM: C.3 Product: Design Attributes

DESCRIPTION:

Importance - Software design can strongly influence cost and schedule. There is a shortage of skilled design personnel and design literature in the area of embedded computer systems. Poor first designs are hard to correct. An example of this is a recent experiment in industry to test the effectiveness of two different programming languages. Independent teams with comparable experience were assigned to the same development project but each used different programming languages. The difference in basic design between the two teams so influenced the effort that no conclusions on the object of the experiments, the language comparison, were possible.

Design concepts and constraints are driven by technology. For example, in the world of "computer plenty" with the use of VHSIC, designs of the software may be driven more by simplicity and ease of understanding than by timing and memory constraints (at least for some applications). A more focused body of literature on software design as it relates to system computational architecture and performance is necessary.

Architectual Considerations - Software design for embedded computer systems is constrained by computational system architecture. Thus, it is important that software design factors influence architectural structure, sizing, and layout. Adequate timing resources, memory, word length, and communication throughput are essential in the system architecture to permit consideration of alternatives and to promote simplicity in software design. As an example of architecturally induced complexity, an engine control system was implemented on an eight bit microprocessor architecture which required double precision to achieve necessary accuracy. The greater precision caused timing and memory problems. Substantial tailoring of algorithms was required, simplicity of design was lost, and testing was severely complicated. The effort became so complicated that it was substantially delayed. The root of the problem was a lack of understanding of the application.

Distributed architectures and special purpose processors require a systematic understanding of both hardware and software design to achieve the desired processing rates. The desire for continued operation when various system components (computer or otherwise) degrade or fail burdens software design with failure recognition, fault isolation, and redundancy management. A better understanding of software design implications of various distributed, fault tolerant, computational architectures is necessary.

The special areas of signal processing and flight controls are examples of the need for understanding of both hardware architecture, fault tolerance, and software design in the environment of rigorous, intricate timing requirements. Software design implications, higher order language utility, and testability all bear on these applications.

Security - Another challenging design problem is posed by security. The system and software implementation must be designed to prevent loss of sensitive security information, to avoid inadvertent compromise of the data, to prevent insertion of deliberate misinformation, and to eliminate sabotage. A few aspects have been studied, such as multilevel secure operating systems, but general design for high reliability and high invulnerability is still not understood.

User Interface - Pilots, maintenance personnel, ground support officers, communications officers, etc., interface with systems through procedures and symbology implemented in software. Well established design requirements for software, which assure the best user-system interface, still don't exist for most applications. The case of the electronic warfare maintenance concept, described elsewhere in this report, exemplifies a trial and error approach to the user-system interface design problem. In the case of pilots, additional function due to automation tends to add to workload. An important objective should be to help reduce this trend through better software design requirements.

Testing will become more important as applications become more complex, and more sophisticated. As all of these requirements come about, software test criteria, test philosophy, and test methods will become key activities.

Faulty Design -- The problem of faulty design of software is a major factor contributing to software development overruns and a major contributor to a system's poor performance during testing. There are many variables that can cause a software design to be faulty or inadequate, though not all faulty designs result in poor performance. Each contributing variable will be mentioned and commented upon.

In a parallel effort, where both software and hardware design are concurrent, misunderstandings occur between hardware and software engineers causing misconceptions on the part of the software personnel as to how the hardware works. In such a case, software development is usually on a different processor than that for which the software is being developed (or a stripped down prototype of the proper processor). When the actual complete version of the processor becomes available, the software architecture invariably requires changes.

During the system definition phase of a development, inadequately defined requirements usually cause software engineers to make assumptions relative to implementation that may or may not be correct. The evolutionary design concept, where a design develops as it is being implemented, probably has been the biggest cause of faulty software design in the past in systems that were implemented using assembly language. In those systems, the architecture of the software and the design of many of the modules are fixed early and become difficult to change or modify at a later time as the requirements evolve. (Hopefully, the use of higher order programming languages will alleviate this potential problem.)

In connection with the above, the architecture of software can lack modularity, i.e., be designed monolithically. This results in a waste of memory since the software must be duplicated each time it is needed rather than calling a common function. The opposite of this is the highly efficient, densely packed software (in assembly language), which has no room for expansion or modification, and in which even the smallest modification causes chaos throughout the system. In both approaches, debugging can prove difficult and time-consuming.

Improper selection of languages can lead to complications in implementations since specific languages were designed for particular applications. For example, the selection of assembly language to perform scientific computations

causes a programmer to generate many lines of code which a compiler could generate from a few lines of FORTRAN statements. The use of COBAL or BASIC statements to generate the code for a real-time I/O interface could result in processes whose run time is greater than the response times required by the I/O.

Programming style also can contribute to faulty design: the use of self-defining rather than labelled constants, the use of implicit verses explicit addressing, and the use of implied verses defined length for variables. This becomes a major problem when personnel change in the middle of a project and another programming style is introduced which is not compatible. Conflicting standards, or incomplete standards for interfaces, can contribute to erroneous assumptions thereby creating erroneous design.

Lastly, differences exist between a "real world environment" and the sterile environment of a computer laboratory.

Constraining Hardware - Many older systems have instruction set architecture constraints that make evolution to an integrated system difficult or impractical. In most instances, the hardware is very old and the software is designed to minimize the impact to the system of the hardware constraints, not to maximize the value and flexibility of the software.

Unrealistic Objectives - Often the original objectives are much too ambitious: a complex system is to be designed, built and installed as one piece. The use of monolithic rather than incremental development often leads to cost overruns and schedule delays. Examples of this approach are rife.

Design for the Complete Life Cycle - Design decisions are rarely made with the complete life cycle of the software in mind. The design should consider the long period of maintenance which is a reality for all weapon systems. Proper design can reduce maintenance costs considerably but this aspect of design is rarely considered. All resources (i.e., hardware, software, personnel qualifications, etc.) needed for life cycle should be clearly stated but rarely are.

Flexibility to Change Different weapon system elements require different response times to anticipate and address software changes. These can either be changes in the threat or to upgrade system performance. System architecture and initial software design strongly influence the ability of software to be modified. Other important factors bearing on flexibility and responsive change include pre-emptive engineering, test requirements, and operational concept, advance intelligence information, the support environment, communication of changes in capability, operator procedure changes, and correlary changes such as training requirements.

The Soviet REC Concept - Early in 1942, the Soviets embarked upon a combined arms concept for the employment of electronic warfare assets. They have developed the concept and fielded the equipment necessary to combine fire power with classic electronic jamming on the battlefield. This concept is referred to as Radio Electronic Combat (REC). This concept targets all NATO and U. S. Command and Control elements (control activity, communication links, and controlled entities). Even while the build-up in REC capabilities has been accomplished, the Air Force has increased its dependence on the electromagnetic spectrum. The fact that these two philosophies are evolving, dictates that an ability to meet changing REC threats is necessary in a timely manner.

Our ability to place ordnance on target is directly tied to our effectiveness in countering the enemy's electronic threats. As changes occur, we must change our fire control elements quickly and accurately. For example, during test of our front line fighters as early as 1976, serious problems were identified (barrage noise, tuned and swept spot, deceptive techniques) which adversely impacted putting radar-directed missiles on target. Findings from the Tactical Air Warfare Center "Green Flag" exercise in 1979 were that deficiencies existed in subsystems, such as terrain following radar (TFR), fire control radars needed electronic counter-counter measures (ECCM) modifications, and that present ECCM alternatives had not yet been adequately exploited. Since that time, the Air Force has expanded the concern to communications systems, bomb-navigation systems, and TACAN systems. Lastly, threat estimates predict that the F-15/F-16 radars and radios were targeted by enemy REC. In addition, these estimates expected JTIDS would also be targeted and the threat would continue to evolve.

Rapid Change - The potential flexibility afforded by digitally controlled systems provides the basic attribute for rapid changes. While a structure has been established to reprogram electronic warfare (EW) assets to meet changing enemy offensive threats in a timely matter, no such emphasis has been focused on systems such as fire control radars to meet changes in enemy defensive techniques. Resources needed to create a capability in this area are: the establishment of a concept of operations for pre-emptive and quick reaction capability activities; access to proper intelligence data; and support capabilities and instrumentation to support the concept.

Turnaround time on changes will depend on the complexity of the system, the software design, the required change, the skill and understanding of the software engineers, and the extensiveness of required testing, not to mention maintaining baseline documentation, configuration control, and field operation information. These elements must be made to interact to reduce the reaction time to the minimum possible, and not to a normal block change time table.

Specific Design - Software developed for embedded computer systems is normally tailored to the specific application and the hardware environment. Requirements levied on the developer frequently constrain the software design to the hardware devices which, in turn, are constrained by the mission needs. One of the most common problems confronting the software designer/developer is the need to package the software to optimize the available computer resources (size, timing, and memory). Hardware trends toward physically smaller devices (VLSI and VHSIC) are providing for faster processing, more memory in a fixed volume, and a propensity to design software modules optimized to these devices. Adding to this the greater sophistication and complexity of today's weapon systems, the tendency is to develop software packages that are system unique, non-transportable, and non-reusable (i.e., immutable software).

The inability to exploit previously developed ECS places the acquisition agencies in the posture of having to pay over and over again for essentially the same software. Major contributions to this problem are: a lack of standard hardware; inadequate documentation of ECS; a lack of standards for software development; and the inadequate sizing, timing, and memory capacity of embedded computer systems. A more generic problem is the inadequate transfer of information about existing and available software resources.

Software immutability can be overcome but it will require a major effort in government and in industry. Reluctance in the program office and in industry to use software developed for another purpose by other people can be expected. There is a need to establish a methodology which addresses the design, development, test and documentation of software specifically designed for reuse.

The most important action to be taken will be to ensure that reuseable software segments are fully documented and easily understood. This will require a careful analysis of the present acquisition methodology to determine changes needed to enforce reusability. Potential items that will be impacted are:

- Defense acquisition regulations
- Military standards and specifications
- Statement of work
- Work breakdown structure
- Data item descriptors
- Solicitation procedures
- Proposal evaluation criteria
- Source selection procedures
- Contract and project incentives

The potential of previously developed software should be addressed as early in the acquisition cycle such as when the mission element need statement is being formulated by the combat developer with the assistance of the material developer. It will be necessary to review the specifications and documentation of old, current, and future systems for common requirements and applications. It is highly likely that software modules exist, or are in development, which implement some of the algorithms, interfaces, processes or conditions cited in the need statement.

Even if the actual software is not reuseable, it is possible that the functional design is directly applicable and could be reused. The material developer should be alert to the potential for moving software between systems. As we proceed with efforts to standardize (e.g., hardware, software, interfaces, documentation, et al), the opportunities for reuse of systems, subsystems, and components will increase, as will the visibility of candidates for reuse.

Identifying reuseable software is severely hampered by the poor quality of the documentation which purportedly describes the functional design, detail design, and coding of software. A major problem today is that the information on other systems does not lend itself to reusability considerations. Specifications are not written in a common language and format, nor are they consistent in the quality of information conveyed. There is a real need for a standard for specifications. Ongoing efforts to produce a single set of DIDs for software specifications will be a significant improvement. Further steps are necessary to de-personalize (both in terms of the system and the writer) specifications to improve understanding. Perhaps a common specification language could be built and provided to the system developers in government and industry.

The key to improving the reuse of software is standardization. However, even if all facets of embedded computer systems were standardized, there will continue to be extenuating conditions and requirements for which one or more standards will be waved. The continued growth and advancement of technology should not be stifled by inflexibility in standards. Standards must be compatible with the rate at which technology is changing. All levels of management must ensure that the standards are adhered to, and that there be positive control of deviations from and changes to these standards. Specific to software reuse is the need for standards in the areas of language, toolsets, interfaces, modularity, design specification, documentation, coding, and the programming environment. A problem that exists (and will persist) is how to ensure the mutual compatibility of the standards. A control structure is required to maintain the integrity of the set of standards and provide assurance that the impact of a change in any standard is considered in all other standards.

Reuse of software will have an impact on the planning, award, performance, and measurement of system and software contracts. The major difficulty is in clearly and precisely documenting the software to be reused: how it will be incorporated; the testing requirements; the data rights; and the conditions under which the software can be changed. This requirement applies to the government, when preparing the procurement package, as well as to the contractor when preparing the proposal. When the software to be reused is government-furnished, problems, with attendant risk,

will arise if the software is not fully documented and the conditions for use are not well defined in the contract. The contract must clearly address the areas of responsibility and liability so as to preclude conflict. When the software to be reused is contractor-furnished, the government must be provided a complete documentation package including test procedures and results as well as a clear definition of any propriety rights claimed.

PROBLEM: C.4 Product: Documentation

DESCRIPTION:

Documentation plays several roles in ECS development and support. During acquisition and support, documentation conveys information on requirements, design and interfaces, implementations, test procedures, and test results. Not only does the documentation convey requirements and design from one group to another, it also serves as milestones for completion of certain activities.

In military systems, where budget varies and fluctuations abound, documentation serves as a record of work completed to date so that it may be possible to resume at a future date. Also, documentation serves as a contractual requirement describing what must be developed as well as a record of what has been developed. Documentation also serves as a baseline against which changes may be assigned so that an orderly and controlled engineering development and modification can take place.

Acquisition - The problems with documentation are legion. Formal documentation is described by data item description (DIDs) which indicate the format and content requirements of each type of document. The numerous DIDs overlap in content. The task of the acquisition agent is to select and modify those DIDs appropriate to the system and its future operational and support requirements. Since formal documentation is usually deemed to be very expensive to acquire and maintain, the acquisition agency may feel pressured to mimimize documentation or, perhaps, to rely on "informal" or "best commercial practice" documentation.

Neither of these last two types of documentation is well defined; hence, the acquisition agent is not sure of what the "paper product" will be like. Often, the agent acquires the wrong type of documentation, documentation poorly written, or documentation written for the wrong level of reader skill or background.

When a project begins to slip behind schedule, the documentation discipline tends to relax and more reliance on oral communication and human memory takes place. This adds to the risk of miscommunication, and increases the cost of subsequent documentation due to the need to reconstruct

forgotten concepts and details. In addition, when funding is reduced, documentation is sometimes one of the first items deferred or deleted by the acquisition agent.

After the documentation for a large system has been prepared, changes to the system (even inexpensive changes to implement) can have substantial impact and cost to update the corresponding documentation that must be fielded to represent the change for training, education, and system or detail definition. Documentation is usually labor intensive. The expense of maintaining the baseline could be lowered by various technology thrusts.

Transition - One of the obstacles to the transition from contractor development and support to internal support is the quality of documentation of the system and system software. Documentation is the mechanism for passing knowledge from one group to another when the work moves from the contractor to another group. Documentation has long been considered by support agencies as a problem area because: data packages are often incomplete; documentation often contains many inconsistencies, errors, or ambiguities; data are withheld as proprietary; there are delays in disseminating current and accurate documentation; and there are, at times, disagreements between the acquisition agent and support agency as to what is required.

Support - During the support phase, documentation serves as a baseline from which modifications and upgrades can be made either through contract or internal staff. However, as during development, modification of documents is a labor intensive task with much room for automation. In one program examined, several man-months of design effort were needed because documentation on an interfacing system was inaccurate. In another case, a subcontracting arrangement between the support system developer and the airborne system processor vendor was required because documentation had not been acquired.

The extent of documentation errors is not easy to pinpoint. It is believed that, in many cases, little effort is expended in "cleaning up" documentation. In one case where documentation was moderately scrutinized, literally hundreds of software documentation errors were

identified ranging from typographical errors to basic inaccuracies. In another example, documentation and support tools for internal support were delayed because a major software package was not specified as deliverable during acquisition.

Documentation Focuses on Details - Much documentation for systems is weak and outdated. In most cases, there is little or no formal documentation in the key areas of software systems design, program design, system interfaces, and correctness analysis. Documentation tends to be of the minute details of program logic and is developed after the fact. Documentation standards often are not evident in the finished documentation.

System Documentation is Limited - System documentation is often inadequate and incomplete. There is no standard method for tracing system requirements through the software development phases, nor to ensure that all design information is accounted for and handled in the appropriate phase. The lack of a complete project history, including design decisions and the analyses that led to them, can waste a lot of time if those decisions are reopened because of a change in requirements or a reevaluation of the system.

D. PEOPLE

The rapid spread of digital technology into weapon systems has and will continue to manifest itself in a shortage of skilled system engineers, software engineers and managers. Advancing technology requires new systems skills and often provides barriers to understanding for the uninitiated. Perceived government-industry pay differentials and the military rotation system aggravate the situation by prompting turnover of experienced personnel and placing new, inexperienced people in responsible positions.

PROBLEM: D.1 People: Skills

DESCRIPTION

General - The skill requirements for ECS system engineers, software engineers and management are interdisciplinary in nature. For example, a software engineering team developing or supporting avionics software must have a functional understanding of radars, inertial navigation units, weapons delivery and control, electronic warfare control, air data computation, the flight environment and the threat, as well as computer and communication architecture, software requirements derivation, software design, algorithms, code, checkout, simulation modeling, integration, and software and system test including support software development, test, and use. Software engineering is not just programming. It is, in fact, systems engineering at both the abstract level and the detailed level. Software engineering is to programming as systems engineering is to drafting. Support of ECS requires the same knowledge and skills as development.

Communications - Different weapon system types require different mixes of engineering and computer science talents. The key skill is that of communication across traditional academic disciplines. The engineer must be able to understand and to be understood by the computer professionals in order to evolve coherent software requirements and designs. The true software engineer is skilled both in traditional engineering and computer science, and provides the interpersonal communications for the multi-disciplinary team. Few universities have developed such an inter-disciplinary approach to software engineering; in fact, few universities teach software engineering.

Currency - One key factor for both technical and management personnel is currency. The transfer of new technology from R&D activities to technical personnel performing ECS tasks is very slow. Management personnel often lack the technical knowledge and currency to manage complex systems development and support activities because, not being directly involved in the technology, they lose direct touch with the rapidly evolving technology.

Government Acquisition/Support - In system acquisition and support, the government needs adequate and mature technical skills to prepare requirements and to monitor

developments. The government also needs managers who understand the acquisition and support process and the cost, schedule, performance, and risk impacts of decisions effecting the software and, hence, the system. Such skills are not learned quickly. The skill shortage tends to manifest itself not so much in major, high priority acquisitions since their priorities command the assembly of the best acquisition talents in the services. (Even high priority acquisitions have had severe problems.) It is much more apparent in smaller programs where less experienced government engineers and managers are assigned high levels of responsibility.

Acquisition is not and cannot be taught in the universities. Graduating software engineers or computer scientists usually want to practice their trade by actually designing, coding, and testing software, thus building their practical experience. They usually have neither the desire nor the qualifications to perform acquisition engineering activities such as preparation and reviews of specifications or designs, or other technical activities. The software engineer must have "hands-on" experience with several systems before supporting an acquisition.

The skills to manage ECS acquisition and support activities rest on a knowledge of general processes of acquisition and support, and how ECS are similar to and different from those processes. Again, there is no formal instruction that is adequate to develope those skills.

PROBLEM: D.2 People: Availability

DISCUSSION:

General - The demand for engineers and computer scientists exceeds the supply. The demand for people who are multi-disciplined in engineering and computer resource areas, or who can communicate effectively in a multi-disciplinary environment, even more sharply exceeds the supply since those skills represent the intersection of skill pools, each of which is in high demand. This demand for skill exists both in industry and government. The shortfall in supply just for programmers has been projected to be 30% by 1985.

An example was cited where the lack of qualified ECS personnel within a system program organization actually degraded test planning and efficiency. The task of reviewing the operational flight program listings and patches fell on test personnel who did not have an understanding of software. Project personnel felt that without this software expertise, it was impossible to plan the detailed aspects of the tests, such as test sortie profiles and to test software changes accurately and thoroughly.

Need - The government personnel shortage is couched in terms of "supply and demand." The correct implication is that the magnitude of the true need is not clearly understood. There are no data on which to base estimates on the number of software acquisition engineers or managers needed for a project to reduce the acquisition risks appropriately. The roles of these professionals must be more clearly defined and related to the risk characteristics of a particular system acquisition.

Policies - The Services have a fundamental policy of rotation of officers to broaden their scope and enhance promotional potential. It is not infrequent (particularly in smaller programs) that this rotation policy replaces a maturing software engineer or manager with one who has not yet had the benefit of acquisition or support experience. Furthermore, the capable personnel in this area will often look for non-ESC assignments to enhance their promotion potential even more.

Shortage of Skilled Personnel - The problems discussed in the Air Force's description of the personnel skill shortage problem are also applicable to Army battlefield automated systems. There is a lack of adequate numbers of trained civilian and military personnel to support in-service requirements. If software support is to be cost effective, there is also the need for a significant percentage of the software development personnel to carry over the support role.

Both the Navy and Marine Corps use the same personnel for development and support. This is not usually the case for Army systems. In general, at least 40% of the resources required for development of software are required for in-service support. Failure to have some of the development personnel available for support could result in increasing this ratio to over 75% of the development cost.

Technical Experience - The Navy has become heavily dependent on the support contractor for many life cycle management activities due to a lack of experienced technical personnel within the Navy. These technical activities typically include: system design and trade-off studies, change control, configuration management, subsystem integration and test, system build and test, verification and validation, quality assurance and control, and training.

Shortage of Qualified Personnel - There is a severe shortage of qualified personnel in many technical areas. There is a tendency to substitute large numbers of unqualified or poorly qualified workers when qualified workers cannot be found. Partly due to the severe personnel shortage, there are problems in attracting and retaining qualified personnel. Staff turnover can be crippling.

Reliance on Contractor - There is unusual reliance within DoD on software contractors, both for creating the software and managing the process. This reliance creates problems when monitoring contractor performance, and is counter-productive to developing a good working knowledge of software. As the percentage of work done by contractors increases, the problems also increase. Turnover of personnel, lower quality of contractors, and long learning curves contribute to the many problems caused by the over-reliance on contractors.

PROBLEM: D.3 People: Incentives

DISCUSSION:

Seller's Market - The environment created by the "sellers market" for digital engineering skills has placed a burden on ECS management in the retention of experienced personnel. While it is generally felt that the majority of ECS engineers leaving the government are migrating to industry, it was found that many in civil service were actually transferring to more desirable jobs in other Government agencies.

Reasons generally perceived as having the most impact on retention were salaries, promotion opportunities, Government "red tape," and lack of opportunity for career development. While it was recognized that some progress had been made in expanding the military's technical personnel requirements, the fact remains that civil service qualification standards do not recognize system engineering or software engineering as a discipline. The present incentives actually encourage change of positions and change of types of work; for example, those in software support look at development as a more glamourous and prestigious activity. Civil service grade structures tend to support that perspective. Those in acquisition tend to view contractor activity as more rewarding.

Career Path - Career paths for engineers and other people in computer related skills in both the military and civilian areas do not possess sufficient growth. Excellent technical people are often promoted out of their fields into management, in spite of the fact that their expertise is sorely needed at the technical level. Military officers in some cases resign and return as contractors or civilians to retain their computer skills and remain with specific systems.

The Air Force software personnel shortage is due in a large part to the fact that there is no formal, effective career management program to attract and retain qualified engineers and managers in the ECS disciplines. An effective career management program must provide for classification and qualification fo standard adjustments, professional and continuing education, training, challenging assignments, a means to track ECS skills, and the identification and communication of career paths for ECS personnel. The "up or out" policy further contributes to this mobility.

Incentives - Work incentives are often reversed: competent personnel are overworked and inadequately challenged, while their pay may be unrelated to competence and productivity; incompetent personnel may be underworked with their pay unrelated to productivity. Because of salary shortcomings, the ability to attract highly qualified people in substantial numbers does not exist. Thus, even if the use of contractors were abolished, the current salary scales would not be sufficient to fill the gap. The current salary structure draws average and some above average people.

Another important reason for problems in career development is the lack of reward for excellence. This is directly attributable to a lack of success criteria within the ECS engineering discipline. Data pertaining to these areas need to be gathered and formulated into accepted standards.

Recruiting - It is perceived that the Government is outbid by industry for the services of more desirable candidates. Reasons given for industry's advantage are higher pay, paid plant visits for prospects, and restrictive civil service hiring practices. Personnel interviews and documentation research cited manpower shortages of up to 40% of that required for task accomplishment. Without exception, those interviewed cite difficulty in filling open positions with experienced, highly qualified personnel, with grade and salary restrictions given as the major cause.

APPENDIX B

Previous Studies on Software Problems

INTRODUCTION

Appendix B contains a brief summary of a number of existing studies on software-related issues. These studies were cited by the task force members as applicable to the effort and include DoD reports as well as reports from all three Services. Software problems identified in each report are referenced via codes (e.g., A.2, B.1 and C.2) to detailed discussions in Appendix A, Problem Taxonomy and Descriptions. The use of a problem taxonomy permits problems to be easily referenced and cross-indexed within all parts of this report.

List of Studies Summarized

	A. LIFE CYCLE					B. ENVIRONMENT				C. PRODUCT				D. PEOPLE		
	REQUIREMENTS	MANAGEMENT	ACQUISITION	PRODUCT ASSURANCE	TRANSITION	DISCIPLINED METHODS	TOOLS	REINVENTION	CAPITAL INVESTMENT	DOESN'T MEET THE NEED	SOFTWARE METRICS	DESIGN ATTRIBUTES	DOCUMENTATION	SKILLS	AVAILABILITY	INCENTIVES
* * NAVY * *	1	2	3	4	5	1	2	3	4	1	2	3	4	1	2	3
1. STI 5/81	*	*	*				*			*		*	*	*	*	*
2. TADSTANDS @81						*						*		*	*	
3. NAVSEA @ 80/81	*	*							*				*	*		*
4. SEATECS 10/15/80									*						*	
5. NCCS 9/30/80		*		*	*	*	*	*	*			*	*			
6. SSF 9/17/80									*							
7. DOD-ISA 3/26/80						*	*		*						*	
8. COMPREP 12/75	*															
9. TCA&E 10/75								*								
10. DOD S/W MGT 6/75		*			*	*			*			*	*	*		
* * ARMY * *																
11. JLC 11/81											*	*	*			
12. PDSS 5/80	*			*	*		*		*						*	
13. DARCOM 11/7/78	*	*	*	*	*		*						*			
14. S/WSYMPOS-78	*						*				*					
* * AIR FORCE * *																
15. PLAN ECS 10/81	*	*	*	*	*	*	*		*			*	*	*	*	*
16. JLC-ENG 10/81															*	*
17. AUD/MGMT ECS 10/80	*		*		*	*				*		*	*		*	
18. ECS-II 9/80	*	*	*	*	*				*			*	*	*	*	*
19. SADWG 7/80	*	*	*				*				*	*	*		*	
20. S/W COST MODEL 6/80											*					
21. S/W REQ-EC 3/80	*	*	*		*								*	*		
22. S/W ADMIN 2/80		*		*	*									*		
23. S/W WORK-79 10/79		*	*				*				*		*			
24. MGMT-79 8/79			*	*	*		*					*	*			
25. COMTEC-2000	*	*		*	*		*					*		*	*	
26 OPER S/W M D 78		*	*		*		*					*		*		*

FIGURE B-1 THE PROBLEM/STUDY MATRIX

B.1 Candidate R&D Thrusts for the Software Technology Initiative, Samuel T. Redwine, Jr., Eric D. Siegel, and Gilbert R. Berglass, May 1981. (STI)

Funded by: Office of the Undersecretary of Defense for Research and Engineering (Electronics and Physical Sciences)

Objectives: "to lay the framework for the process of identifying, assessing, selecting, and initiating research and development thrust to solve problems common to all services and DOD components."

Methodology: The candidate research and development thrusts were compiled from recommendations in published reports and from suggestions submitted to DOD in response to public announcements of the Software Technology Initiative.

Structure: Five sections: Introduction, Short-Term Payoffs, Medium-Term Payoffs, Long-Term Payoffs, and Summary. 6 Appendicies: Descriptions of Candidates, Other ideas, Software Problem Areas, Summary of Some Reviewed Studies, Evaluation Considerations, and Software Initiative Questionnaire.

Problem Areas: A.1, A.2, A.3, B.2, C.1, C.3, C.4, D.1, D.2, D.3

Details of Problems Addressed

A.1 Requirements: Lack of usable, comprehensible, measureable, validated requirements for goals; inappropriate levels of detail.

A.2 Management: Lack of good models for schedules, budgets, and sizing; inadequate cost and schedul monitoring.

A.3 Acquisition: Lack of standard methodology; inappropriate construct type; management of trivial details from high levels of authority; rights not acquired to software embedded in system; rights not acquired to software tools used to develop software.

B.2 Tools: Flawed and conflicting standards; software tools that are: unavailable, unpublicized, hard to use, inefficient, and unevaluated; poor test environment.

C.1 Doesn't Meet the Need: Lack of a "lessons feedback loop to the software development community; lack of communication between users and implementers.

C.3 Design Attributes: Inappropriate hardware constraints; performance goals too high.

C.4 Documentation: Lack of a project history with design analyses and decisions; lack of phase-to-phase continuity (traceability).

D.1 Skills: Poor design skills; weak project leadership and coordination; unsuitable measures of competence; poor training.

D.2 Availability: Shortage of qualified personnel and a tendency to substitute poorly qualified personnel; high turnover.

D.3 Incentives: Incorrect job grades and descriptions; no career paths, pay unrelated to competence.

B.2 Chief of Naval Material Task Force on Tactical Digital Standards, undated draft, estimated @ 1981. (TADSTANDS)

Funded by: Office of the Chief of Naval Material

Objectives: Analysis of advantages and disadvantages of the NAVMAT Tactical Digital Standards on Computers and Software.

Methodology: N/A

Structure: The report of the task force is in chart form as delivered in January, 1982.

Problem Areas: B.1, C.3, D.1, D.2

Details of Problems Addressed

B.1 Disciplined Methods: Standards not adequately and uniformly applied; in-process verification of standards is spotty; no accountability of project managers who ultimately back off standards; contractors convince managers to try non-standard approaches.

C.3 Design Attributes: Request For Proposals (RFPs) don't address design features which facilitate change and re-use.

D.1 Skills: Project managers lack knowledge and experiences to properly manage software.

D.2 Availability: Current and growing shortage of software engineering and management personnel.

B.3 Assessment of Naval Weapons Software Systems: Initial Analysis of Eighteen Software Systems, Bennet P. Lienta of UCLA and Peter Wegner of Brown University, 1980. (NAVSEA)

Funded by: Assistant Secretary of Navy (RE&S)

Objectives: "to assess the current state of Navy computer systems in order to provide a basis for improving their cost, reliability, and performance."

Methodology: Analysis based on data from eighteen Naval and Marine Corps computer systems at eight sites. A data collection form was developed to serve both as a means for gathering information about Naval computer systems and as a vehicle for discussing computer systems with installation managers, designers, maintainers, and implementers.

Structure: Five chapters: Introduction and Scope of the Study, Methodology of Data Collection, Tabular Summary, Analysis Results, and Conclusions and Recommendations. Appendix: Data Collection Form.

Problem Areas: A.1, A.2, B.4, C.4, D.1, D.3

Details of Problems Addressed

A.1 Requirements: Inability to verify requirements; correlation of requirements to design; control of requirements.

A.2 Management: Insufficient Quality Assurance (QA) and Configuration Management (CM); lack of tracking systems.

B.4 Capital Investment: Insufficient timesharing base; insufficient automated support.

C.4 Documentation: Lack of adequate user and training documentation; inadequate application of standards.

D.1 Skills: Long learning curves; turnover in personnel.

D.3 Incentives: Salaries and career paths of military and civil servants inadequate.

B.4 Software Engineering Automation for Tactical Embedded Computer Systems (SEATECS): Functional Analysis, Systems Consultants, Inc., October 15, 1980. (SEATECS)

Funded by: Naval Ocean Systems Center

Objectives: This report documents the results of a functional analysis of the Fleet Combat Direction Systems Support Activity at San Diego (FCDSSASD), capacity to develop and support computer programs for embedded computer systems.

Methodology: A model was developed based on the current and projected (through 1990) mission responsibilities and resources.

Structure: Six sections: Introduction, FCDSSASD Mission and Functions, Functional Process Analysis, TECS Workload Analysis, FCDSSASD Resources Analysis, and Conclusions.

Problem Areas: B.4, D.2

Details of Problems Addressed

B.4 Capital Investment: Insufficient computer resources to meet projected workload; inadequate test equipment; inadequate tools.

D.2 Availability: Insufficient manpower of required skill levels to meet projected workload.

B.5 NCCS Ashore Software Audit, Lesko/Fox Associates, (now Software A&E), September 30, 1980. (NCCS Audit)

Funded by: PME 108 (now PME 120)

Objectives: The audit team evaluated the software of the four major subsystems of NCCS Ashore. The objectives were to evaluate the quality of the software from a management and software engineering point of view, to evaluate the limits on evolution of each subsystem, and to evaluate the life cycle management and support facilities of each subsystem.

Methodology: The audit team was composed of industry, university, and Naval personnel. Interviews of PME 108 personnel, support site visits, and document reviews.

Structure: Six sections: Executive Summary, History and Overview, Objectives, Findings, Subsystem Evaluations and Recommendations, and NCCS Ashore Recommendations. An NCCS Inventory is included as an appendix.

Problem Areas: A.2, A.4, A.5, B.1, B.2, B.3, B.4, C.3, C.4

Details of Problems Addressed

A.2 Management: Life cycle planning and management is ad hoc; no means of measurement and effective control.

A.4 Product Assurance: Testing ad hoc; no tools or procedures for system evaluation, e.g. performance monitors.

A.5 Transition: Great variance in standards applied; standards inconsistently applied.

B.1 Disciplined Methods: Great diversity in software engineering practices; ad hoc techniques.

B.2 Tools: Poorly supported by automated tools.

B.3 Reinvention: Same functions reimplemented differently and supported for each subsystem.

B.4 Capital Investment: Hosting on constrained target computers; insufficient computer power.

C.3 Design Attributes: Software not well designed for change.

C.4 Documentation: Insufficient documentation; non-standard; out of date.

B.6 Software Support Facility Investigation, September 1980. (SSF)

Funded by: PME 108-5 (now PME 120-3)

Objectives: To summarize the projected NAVALEX software development and maintenance workload through 1992.

Methodology: "A literature search was conducted to identify documented expectations concerning the evolution of hardware and software technology during the study time frame. A survey was conducted by interviewing representatives from government, industry, and academe. Survey instruments were designed to elicit information necessary for the study."

Structure: Two reports: "Software Support Facility Investigation: Technology Survey," September 19, 1980 and "Software Support Facility: Cost and Facilities Study," September 16, 1981.

Problem Areas: B.4

Details of Problems Addressed

B.4 Capital Investment: Insufficient computer resource; inadequate tools.

B.7 Report of the Department of Defense Instruction Set Architecture Standardization Panel to the Management Steering Committee for Embedded Computer Resources, March 26, 1980. (DOD-ISA)

Funded by: Office of the Undersecretary of Defense (Research and Engineering)

Objectives: In November 1978, the Undersecretary of Defense, Research and Engineering asked that a panel with DOD component representation examine the question of proliferation of instruction set architectutes (ISAs) in DOD systems.

Methodology: A panel was formed, studied the question, sought the advice of industry, and recommended standardization on a limited number of instruction set architectures be adopted as DOD policy.

Structure: Three sections: Executive Summary, History and Background, and Recommendations and Rationale. The last section contains three recommendations: Proposed Instruction, Life Cycle Cost Model, and ISA Development for Large Word Size Computer.

Problem Areas: B.2, D.2

Details of Problems Addressed

B.2 Tools: Use of assembly language and non-standard High Order Languages (HOLs) for implementation. Proliferation of ISA's leads to need for multiple support tools.

D.2 Availability: Shortage of skilled personnel exacerbated by proliferation of ISA's.

B.8 Report on COMPREP (Composit Reporting) System, CNO Project X/C 13, 12/75. (COMPREP)

Funded by: Chief of Naval Operations

Objectives: Evaluation of the COMPREP system

Methodology: Operational testing in four Atlantic Fleet Ships in April - May 1975

Structure: Memo with attachments

Problem Areas: A.1

Details of Problems Addressed

A.1 Requirements: Requirements defined by computer specialists rather than ultimate user; system does not meet the need.

B.9 A Survey of Navy Tactical Computer Applications and Executives, Dorian Punj, Stuart E. Madnick, John D. DeTreville, MIT Sloan School of Management, October 1975. (TCA&E)

Funded by: Naval Air Systems Command
Naval Electronics Systems Command

Objectives: This document is the final report of a study of current Navy operating system milieu. It includes information about Naval tactical computer applications, and in particular, information on the major executives used in these applications. Analysis and evaluations of the various operating systems are presented together with implications that these systems have on the design of a family of operating systems for future Navy tactical systems.

Methodology: Visits to Naval installations and facilities to discuss current and future computer applications. Analysis of large numbers of documents concerned with Navy tactical executives.

Structure: Four parts: General Characteristics of Navy Tactical Applications, Major Executives Used in the Navy, Executives for Airborne Applications, and Executives for Shipboard Applications.

Problem Areas: B.3

Details of Problems Addressed

B.3 Reinvention: Proliferation of operating systems; reimplementation of like functions.

B.10 DoD Weapon System Software Management Study, Johns Hopkins University, Applied Physics Laboratory, (June 1975). (DOD S/W MGMT)

Funded by: Office of the Assistant Secretary of Defense (Installations and Logistics)

Objectives: "to identify and define (1) the nature of the critical software problems facing the DoD, (2) the principle factors contributing to the problems, (3) the high payoff areas and alternatives available, and (4) the management instruments and policies that are needed to define and bound the function, responsibilities and mission areas of weapon systems software management."

Methodology: Review of ten major previous DoD studies, reviews of ten Navy and two Army Weapon Systems, and summaries of discussions with service organizations and industrial software system contractors.

Structure: Eight sections: Introduction, Summary of Recommended Actions, Review of Previous Studies, Highlights of Weapon System Studies, Supplemental Discussions with Service and Industrial Organizations, Discussion of Recommendations, Subjects for Further Investigation, and Guide to Appendices.

Problem Areas: A.2, A.5, B.1, B.4, C.3, C.4, D.1

Details of Problems Addressed

A.2 Management: Insufficient understanding by managers; lack of software visibility; lack of estimation techniques; inadequate cost and schedule monitoring.

A.5 Transition: Unspecified maintenance provisions; high cost of transition due to inadequate provisions.

B.1 Disciplined Methods: Lack of software engineering techniques; non-standard languages.

B.4 Capital Investment: Support systems not established.

C.3 Design Attributes: Inappropriate choice of hardware; insufficient hardware capacity; inappropriate hardware/software functional division.

Lack of requirements for adaptability; inflexible design; non-modular software architecture.

C.4 Documentation: Inadequate documentation.

D.1 Skills: Insufficient understanding by managers.

B.11 Proceedings of the Joint Logistics Commanders Joint Policy Coordinating Group or Computer Resources Management, Computer Software Management Subgroup, November 1, 1981. (JLC)

Funded by: DoD Internal

Objective: To: 1) Provide recommendations to the JLC for (a) developing project management guidelines for selection of software documentation, (b) the addition, deletion or modification of documents in the JLC-list of software documents, (c) implementing the standard set of software documents (JLC-list) within OSD/JLC/Services, and (d) clarifying the relationship between the DoD acquisition life-cycle (milestones, phases) and the JLC-list of software documents (TAB A, pg. 2); 2) Examine the general problem of hardware/software/firmware CI/CPCI partitioning and specification and develop a set of criteria to aid in the selection and documentation process (TAB B); 3) Evaluate the potential for utilizing accreditation of computer architectures as a viable tri-service computer acquisition strategy and determine the role and impact of MIL-STD-175ØA, MIL-STD-1862 and DoDI 5ØØØ.54 within the overall defense computer acquisition standards (TAB C); 4) Evaluate existing software cost estimating models and recommending a tri-service approach to improve software cost estimating methodology (TAB D, pg. 1); and 5) Evaluate whether reusability represents a potentially valuable concept to reduce cost and elapsed time to develop embedded computer system software; (if so) what barriers must be overcome and how does the program manager and/or software manager make reusability a reality.

Methodology: Ninety-two individuals from Government and industry participated in a Software Workshop, 22 thru 25 June 1981. Workshop participants were divided into five panels - one for each of the objectives listed above. Study was a continuation of the effort initiated during the first software workshop held in April 1979.

Structure: TAB A - Report of the Panel on Software Documentation.

TAB B - Report of the Panel on Hardware/ Software/ Firmware Configuration Item Selection Criteria.

TAB C - Report of the Panel on Standardization and Accreditation of Computer Architecture.

TAB D - Report of the Panel on Estimating Software Cost.

TAB E - Report of the Panel on Software Reusability.

Problem Areas: C.2, C.3, C.4,

Details of Problem Addressed

C.2 Software Metrics:

a. Software Cost Estimating - "The panel believes that no existing Software Cost Estimation (SCE) model is sufficient to adapt as an embedded computer system standard." (Tab D - p. 36)

"The panel believes that a judicious use of SCE models and methodology can improve the acquisition and management of software." (Tab D - p.36)

"The panel believes that an improved software cost estimating methodology must be supported by gathering and maintenance of an accurate, complete and coherent database." (Tab D - p. 37)

"The panel recommends that JLC sponsor a program to implement an improved SCE methodology." (Tab D - p. 37)

b. Hardware/Software/Firmware Tradeoffs - The study examines the problem of hardware/

software/firmware CI/CPCI partitioning and specification and develops a set of criteria to aid in the selection and documentation process, particularly as applied to firmware.

Basic questions addressed include: the allocation of functions to firmware, the firmware documentation, the firmware maintenance and support philosophy and the applicable maintenance and control procedures.

There is an underlying question as to whether firmware is a subset of software or a separate entity. Firmware which is "hardware intensive" may be treated as a CI while "software intensive" firmware could be covered as a CPCI. There is a grey area which should be treated as proposed as a CP/H (Computer Program in Hardware) subset of a CI.

C.3 Design Attributes: "In general, it was agreed that a much greater degree of reusability of software in embedded systems is achievable. Current existing software cannot be expected to see much reuse, but in the future, new software can be built for reuse." (Tab E - p. 5)

C.4 Documentation: "The JLC-JPCGCRM-CSM should publish a document selection guidebook for use by acquisition managers." "This guideline should address both software and firmware." (Tab A - p. 17)

"Integrate revision of DIDs with the ongoing development of a MIL STD for Embedded Computer Systems." (Tab A - p. 17)

"Initiate an effort to investigate the ramifications of Ada and other innovations on software development and documentation." (Tab A - p. 18)

"The life cycle should be tailored to specific projects and phases of development." (Tab A - p. 18)

"Retitle software DIDs as software/firmware" (Tab A - p. 17)

"Standards, instructions, directives, guidebook, training criteria, regulations and contractual vehicles must be revised to identify the changes which must be made to include the concept of reusability. The JLC should foster preparation of new policy documents to encourage and enforce reusability." (Tab E - p. 38)

"Exclusive use of higher order languages is a single very strong necessary precedent for reuse." (Tab E - p. 39)

"Research and development of new support tools specifically to provide reusability are needed." (Tab E - p. 39)

B.12 Post-Deployment Software Support Concept Plan For Battlefield Automated Systems, May 1980. (PDSS)

Funded by: DARCOM

Objectives: Develop an Army-wide concept to achieve post-deployment software support of Army Battlefield Automated Systems.

Methodology: Task force of approximately 100 representatives from Army agencies was formed to research documents, conduct interviews, inspect facilities, and form working groups to address issues. Questionnaires were distributed and telephone surveys were conducted to collect systems characteristics and PDSS planning and requirements data. Core group of CENTACS representatives published concept plan report.

Structure: SECTION 1 - Introduction of PDSS Study and PDSS Concept Plan

SECTION 2 - Findings of the PDSS Study

SECTION 3 - Software system structure and a software support model for PDSS

SECTION 4 - Alternative approaches for PDSS

SECTION 5 - Recommended PDSS approach

SECTION 6 - Gross estimates of resources needed (To be identified separately).

Problem Areas: A.1, A.4, A.5, B.2, B.4, D.2

Details of Problems Adressed

A.1 Requirements: The report discusses the need for a single integrated set of procedures, guidelines, and standards uniformly applied to all BAS's including systems interoperability.

A.4 Product Assurance: The report discusses requirements for testing for PDSS, including certification

test, verification and validation tests, PDSS center debug, and integration tests and product assurance tests. It discusses the impact of unnecessary testing delays in fielding revised software versions.

A.5 Transition: The report discusses the problems of achieving effective Post Deployment Software Support (PDSS) by actions taken in the development phase and the need to plan for and build in supportability early in the development and acquisition phases.

B.2 Tools: The report discusses the impact of proliferation of computer languages in current BAS's, prior to the adoption of standard HOL's, including Ada.

B.4 Capital Investment: A separate report discusses the capital requirements for establishment of Software Support Centers (SSC) for Army BAS's.

D.2 Availability: The report discusses the lack of adequate numbers of trained civilian and military personnel to support PDSS requirements and training needs.

B.13 Proceedings of the DARCOM Tactical Computer Software Conference, hosted by the U.S. Army Communications Research and Development Command (CORADCOM) Fort Monmouth, New Jersey, November 7, 1978. (DARCOM)

Funded by: DARCOM

Objectives: To bring together DARCOM elements engaged in acquiring, developing, using and managing tactical computer software for the purpose of discussing issues and problems and enabling the exchange of information and ideas. The conference was oriented to the Project Manager and his set of problems.

Methodology: The three day conference was organized into the following broad topic areas:

- Tactical Computer Software Problems as seen by DARCOM, the Project Manager and the User.
- Software Acquisition and the Life Cycle.
- Acquisition Issues.
- Development Issues.
- Testing and Evaluation.
- Military Computer Family.
- Interoperability.
- Post Deployment Support.

The eight sessions included 25 individual papers and time was allotted at each session for group discussion. A technically cognizant civilian and a general officer were selected to co-chair each session to provide a reasonable mixture of technical and management expertise in addressing each area. A summary session was provided for General Guthrie, CG DARCOM, during which the general officers presented a summary of issues raised during each session. The conference was concluded with remarks by General Guthrie assessing the summaries and conclusions for each session.

Structure: The proceedings of the conference are presented in two volumes, with the first volume including a preface and conference agenda and the results of the first five sessions. The second volume includes the results of sessions 6, 7 and 8 and the chairman's summary add conclusions for each session as well as concluding remarks by General Guthrie.

Problem Areas: A.1, A.2, A.3, A.4, A.5, B.2, C.4

Details of Problem Addressed

A.1 Requirements: Inter-operability in existing systems can only be satisfied by costly modifications and fixes. Inter-operability requirements can be satisfied if:

- interface requirements are clearly stated
- inter-operability concepts are well understood
- interface design planning is conducted on the basis of a total integrated structure

Inter-operability requirements must address some highly complex problems in network design. (pp. 491-500)

A.2 Management: Problems facing the project manager and DARCOM include a lack of overall basic policy, procedures and standards to meet established objectives. Issues include managing and controlling the software life cycle and the role of participants in software development and acquisition policy and guidance.

A.3 Acquisition: Acquisition issues include acquisition management, verification and validation of tactical systems and software development in a joint program.

A.4 Product Assurance: New approaches are presented in support of tactical software test and evaluation as seen from the project manager and the TECOM (Test and Evaluation Command) viewpoint, including considerations for IV&V. (pp. 305-345)

A.5 Transition: Substantial support problems in automated systems result from actions taken in the development process. Issues include post deployment development and testing, software logistics and post deployment software planning activity. (pp. 691-701)

B.2 Tools: Issues addressed include the development of the DoD-1 compiler (Ada) and associated tools to promote the development of the standard military computer family. (pp. 453-459)

C.4 Documentation: Issues in the software development process include the lack of completeness and traceability of documentation for software from the requirements to coding.

B.14 Second U.S. Army Software Symposium, Williamsburg, Virginia, sponsored by U.S. Army Computer Systems Command via the Army Integrated Software Research and Development Working Group (ISRAD), October 25, 1978. (S/W SYMPOS-78)

Funded by: ISRAD, USACSC, DARCOM

Objectives: To provide a forum for information interchange on software research and Development for members of the ISRAD community. To present recent developments in software R&D programs, including those areas which have received little or no emphasis in current programs. To develop an ISRAD community awareness and visibility to these programs and to provide access to the ISRAD R&D programs.

Methodology: The agenda was developed for the two day symposium over a 14 month period by the Symposium Organizing Committee of ISRAD. The symposium was tailored to highlight the DoD Software R&D Technology Plan (September 1977) and included program areas not in the Technology Plan, such as security, human factors and graphics. The Symposium was organized into 18 plenary and parallel technical sessions, at which 48 papers were presented.

Structure: The proceedings of the two day symposium includes a session summary and most of the papers presented for each of the 18 sessions. The proceedings include an executive summary of findings of the symposium and a summary of recommendations.

Problem Areas: A.1, B.2, C.2

Details of Problem Addressed

A.1 Requirements:

a. Security Issues - "The problem of ensuring the security of software, and the operational systems they support,

requires the development of secure operating systems. In military systems there is also a requirement for protection of classified information and privacy data."

"Secure software systems must be certified to verify the multi-level security properties and to prove the design is secure. The goal of a security system is to provide strong assurance that it is impossible for and unprivileged user to compromise protected information." (pp. 107-146)

b. Lack of Inter-operability - Inter-operability between systems requires a clear statement of interface requirements and understanding of inter-operability concepts. Interface design planning must be conducted on a totally integrated structure basis. Software inter-operability considerations are involved in the following: man-machine interfaces, software versus firmware, software inter-operability standards, operator inter-operability training, provision of adequate multi-level security for joint ARMY/NATO inter-operability, considerations for continuity of operations and survivability of Battlefield Automated Systems. (pp. 379-389)

B.2 Tools: Software support tools in embedded computer systems have been plagued with a number of problems such as:

- Non-reusability of existing tools because current systems have unique combinations of language and host target processors.

- Resource constraints such that a minimum number of essential tools are developed for each project. Tools are not the focus during development (the embedded system is), therefore a significant lower level of quality results.

C.2 Software Metrics: The designer of embedded computer systems has the option with today's technology, to allocate logic functions among the hardware, software and firmware of a system. This design opportunity introduces concerns which must be addressed early in the system life cycle. Achieving the most cost effective design will require a thorough trade-off analysis of the technological, acquisition, operational and support aspects of the system. (pp. 460-502)

B.15 Long Range Plan for Embedded Computer Systems Support, TRW for AFLC/LO, October 1981. (PLAN ECS)

Funded by: AFLC/LOEC

Objectives: Develop a long range plan for improving the AFLC embedded computer system support posture for the 1980's.

Methodology: The study, accomplished by a task force of TRW personnel, expanded on the framework established in Phase II of the study which had: 1) established the current baseline of support for the five categories; 2) assessed and forecast major technology impacts and investigated selected support issues. From this material, 12 support objectives were developed and initiatives developed to satisfy those objectives. From those initiatives, the long range plan was developed reflecting recommended implementation for both administrative and programmatic initiatives.

Structure: Volume I--Executive Overview, Volume II--Long Range Plan

Problem Areas: A.1, A.2, A.3, A.4, A.5, B.1, B.2, B.4, C.3, C.4, D.1, D.2, D.3

Details of Problems Addressed

A.1 Requirements: Requirements baseline effects test requirements; suggests ATD change funded concurrently with weapon system change, preemptive engineering problems; investigate and maintenance on AISF specification tools; lays out plan to provide adversary threat changes within fire control elements.

A.2 Management: Recommends status accounting tools; stresses system engineering aspects of ECS support.

A.3. Acquisition: Recommendations associated with funding issues; funding relationship with other initiatives.

A.4 Product Assurance: Recommends IV&V as a multi-purpose activity.

A.5 Transition: Recommendations relating to data quality; IV&V as transition tool; emphasizes problems associated with divided support responsibilities.

B.1 Disciplined Methods: Recommends architectures and standards for multi-use ECS.

B.2 Tools: Analysis is labor intensive; automate documentation and distribution tools; recommends automation and standardization of ECS support processes; recommends MIS.

B.4 Capital Investment: Addresses common ECS support components; limit support system proliferation; recommends ECS support networks.

C.3 Design Attributes: Stresses design for testability; stresses modularity; proposes a modular integrated support facility.

C.4 Documentation: Automate documentation; standardize CM; makes several recommendations relative to configuration management.

D.1 Skills: Need exceeds supply while engineers tasked with non-technical tasks; makes training and professional education recommendations, recommends skill centers.

D.2 Availability: Improve productivity to compensate; suggests matrix organization; several recommendations to improve personnel acquisition and retention.

D.3 Incentives: Recommends ECS career progression.

B.16 Study of Civilian Engineer Recruitment, Retention and Use Through-out The Joint Logistics Commands, 30 October 1981. (JLC-ENG)

Funded by: Joint Logistics Commanders

Objectives: The objectives of this study were to develop solutions and recommendations for assessing engineering manpower needs, recruiting sufficient numbers and quality of engineers, and encouraging maximum retention and optimum use. The study was intended to prove or refute the following perceptions.

- JLC Commands are experiencing recruiting difficulties because federal salaries are not competitive with industry.
- The recruiting programs and methods need to be improved.
- JLC Commands are experiencing a serious retention problem at all grade levels because of non-competitive salaries, job dissatisfaction and lack of promotion opportunities.
- Throughout the JLC Commands engineers were not having full use made of their professional engineering skills, knowledge and abilities.

Methodology: To gather the supportive data, the study group reviewed appropriate literature, conducted interviews with college placement and personnel management officials, and administered an employee questionaire to about 1,700 of the 37,980 JLC civilian engineers.

Structure: The report contains an executive overview which summarizes the conclusions while the main body of the report summarizes JLC Commands demographics, occupations, salaries and benefits, planning for re-

cruitment, recruitment, retention, use of engineers, and supportive data. The data are partially analyzed and numerous recommendations are made.

Problem Areas: D.2, D.3

Details of Problems Addressed

D.2 Availability:

a. Engineering Shortage - "Currently, engineering shortages exist in the computer sciences and most engineering fields at all degree levels. These shortages are expected to decline gradually until 1990, at which time the numbers of new engineering baccalaureates should be adequate to satisfy projected demand for their services. Continued shortages in computer sciences are expected to continue into the 1990s. However, easing of the engineering shortage by 1990 is dependent on colleges and universities having the capacity to educate all students at both the undergraduate and graduate levels who want to obtain degrees and who are judged by those institutions to be qualified to do so (i.e., secondary schools have reduced educational standards and requirements, and have shortages of mathematics and science teachers, thereby depriving college entrants of the academic background necessary for engineering degrees). Increasing enrollments, falling levels of Ph.Ds, faculty shortages and inadequate/obsolete laboratory facilities and apparatus may restrict colleges from training all qualified applicants. If such restrictions occur, there may be fewer engineers available in 1990 than the projections indicate, possibly resulting in continuing tight markets in most engineering specialties and perhaps serious shortages in some of them. Nationwide the 93,500 annual engineering vacancies cannot be fully met by graduating engineers, immigrant engineers and graduates from related fields. A 16,500 annual shortfall will remain until 1990." (pp. 23,4)

b. Recruitment - Current methods of recruitment for "new degreed" engineers are summarized. Discussions with college placement personnel suggest improvements in federal advertising. The most emphasized factor for new engineers was the $3000 (for G-S-7), and $4000 to $6000 (for G-S-5) annual starting salary disadvantage the federal government has in relation to industry. New graduates must perceive that government compensations are "fair" and comparable to industry.

Recruiting from all sources (both internal and external to the government) and at all levels of experience is recommended. (pp. 25-30)

D.3 Incentives

a. Retention Rates - Statistics for 1979 show 16.7% of G-S-5 engineers and 14.2% of G-S-7 engineers are leaving JLC with the majority leaving federal service. These engineers cited a need for more meaningful and interesting work, professional education and cross training, and assignment of a mentor or sponsor.

For all JLC engineers the loss rate for 1979 was only 7.6 % which implies that engineers are not leaving at all grade levels in the same high proportion, a fact that was not anticipated before the study. (pp. 7, 33, 36)

b. Use of Engineers - The report claims that engineers are not fully used on engineering activities, but the extent to which they are being used is clouded by conflicting data. For example, from one element of the JLC survey data the conclusion is drawn that "more than 50% of the engineers spend 50% or less of their time on engineering," whereas from other data in the same survey more than 60% of the engineers indicated that the engineering work they were expected to do kept them occupied most or all of the time. (pp. 43-49)

c. Conflicting Perceptions - "Perceptions of the JLC engineers, as reflected in responses to the questionnaire, indicated that many of the factors which cause an employee to stay with an employer exist in the JLC Commands. A majority of the engineers perceived: (1) their work is challenging; (2) they are treated as professionals; (3) they are held responsible for technical decisions, and these decisions are supported by their supervisors; (4) the risk they take in making technical decisions is appropriate for the potential consequences; (5) maintenance of their technical skills is encouraged; and (6) initiative and innovation in their work are encouraged.

The engineers also perceived a negative side to JLC employment: (1) many of the jobs they are given do not provide them the opportunity to work at or near their full potential as an engineer or engineering manager; (2) they are paid less than their counterparts in private industry; (3) educational opportunities and career progression are limited; (4) the environment in which they work is minimally creative; (5) the rules and management systems at their installation are often not conducive to engineering accomplishment; and, (6) there are too few engineers in their organization. These perceptions may be damaging to JLC recruitment efforts. More than half of the JLC engineers would advise graduating engineers to seek jobs in the private sector rather than the federal government. This confusing picture must be examined and a more accurage analysis made of engineer's perceptions." (pp. 6,7)

B.17 Summary Report of Audit, Management of Embedded Computer Systems, October 1980. (AUD/MGMT ECS)

Funded by: Air Force Audit Agency - OPR is HQ USAF/RD (This is a paraphrased version of the audit findings and management comments.)

Objectives: Review Air Force and MAJCOM guidance for the acquistion of ECSs including requirements definition for hardware, software, supporting documentation, and computer capacity sizing. Acquistion policies concerning the identification of ECS, configuration management, and test planning were also assessed.

Determine the implementation status of the Air Force inventory or ECS hardware and software including an assessment of the Computer Program Identification Number System.

Evaluate Air Force and MAJCOM procedures for logistics support of ECSs to include depot level support of hardware and software, management of Federal Stock Group 70, and initial provisioning for ECS hardware. Additionally, policies relative to the classification of embedded computers and the operation of integrated support facilities were reviewed.

Methodology: Audit work was accomplished at three AFLC air logistics centers and two AFSC buying divisions. The audit was initiated in September 1979, an advance notice of findings issued in February 1980, and a draft report issued in June 1980.

Structure: The summary report of audit consists of a single volume. In addition, ten reports addressing local audits supporting this effort exist.

Problem Areas: A.1, A.3, A.5, B.1, C.1, C.3, C.4, D.2

Details of Problems Addressed

A.1 Requirements: Memory, timing and I/O channel capacities were not adequately considered, documented or monitored during system acquisition. Existing methods for estimating software size and complexity are informal and illdefined. (pp. 22-25)

Post Deployment Software Support concepts need to be reassessed periodically to assure optimum method is being used (organic, contract, combination). (pp. 41-44)

A.3 Acquisition: General purpose computers used in AFLC Integration Support Facilities were acquired and managed per 800 series directives versus 300 series directives. Concern over whether this is proper interpretation of "gray zone" between 300 and 800 directives. (pp. 17-21)

A.5 Transition: Inadequate policy for computer program configuration items identification and configuration management.

B.1 Disciplined Methods: Some software products were not identified as computer program configuration items and their configuration properly managed. (pp. 5-9)

C.1 Doesn't Meet The Need: Work stoppages at an Air Logistics Center (ALC) occurred due to deficiencies in application software. In another case, ineffective interface procedures resulted in test software which did not address the multiple configurations of hardware to be tested. Software delivered to the ALC would not validate the serviceability of any of the hardware items. (pp. 6,34)

C.3 Design Attributes:

Security - Emanation of electronic signals outside several support facilities were not adequately evaluated. Physical security to limit access to facilities needed attention. (p. 19)

C.4 Documentation: Failure to identify some software as computer program configuration items resulted in the lack of documentation and engineering data required to maintain the software. Depot organic support capability was delayed one year and may be delayed two additional years. (p. 6)

D.2 Availability: One integration support facility experienced personnel turnover of 50%. Skills needed in area of engineering and automatic data processing equipment. (p. 42)

B.18 A Study of Embedded Computer Support Phase II, TRW for AFLC/LOEC, September 1980. (ECS-II)

Funded by: AFLC/LOEC

Objectives: The purpose of this study is to develop a long range plan for use by HQ AFLC to manage and maintain Embedded Computer Systems on a command-wide basis in the 1980's.

- Establish a baseline for current ECS support functions and requirements
- Assess and forecast major technology impacts on future systems and their attendant support requirements
- Investigate the potential use of networking and the National Software Works (NSW) and other support concepts

Methodology: Volume II of the report was developed by a working group within TRW. This group, called the Operations and Support Working Group, consisted primarily of TRW field site managers and other engineers and scientists collocated with AFLC engineering divisions at various Air Logistics Centers. This group wrote a series of "white papers" on selected topics which were identified and coordinated as major problems and issues in the support of ECS. Volumes III through VII, and Volume IX were developed by principal investigators assigned to particular ECS category or to the national software works investigation. Volume VIII was accomplished by forming a Technology Working Group of senior engineers and scientists with expertise in the technology areas pertinent to the AFLC ECS support role. Editorial conferences were held within groups. The study lasted from September 1979 to September 1980.

Structure: Volume I -- Executive Overview; Volume II -- Selected ECS Support Issues; Volume III-VII -- Requirements Baselines; Volume VIII -- ECS Technology Forecasts; Volume IX -- National Software Works Investigation.

Problem Areas: A.1, A.2, A.3, A.4, A.5, B.4, C.3, C.4, D.1, D.2, D.3

Details of Problem Adressed

A.1 Requirements: Problems with concurrency and fidelity of Air Crew training devices; discusses problems and a suggested approach to meet adversary threat changes within ECS for fire control elements. Complexity and programmatic issues associated with requirements.

A.2 Management: Discusses the system aspects of ECS software and the changing role of support agencies with respect to ECS support.

A.3 Acquisition: Addresses the problems of funding, but primarily in the support area; split-funding for modifications and firmware funding; ADPE versus ECS computer acquisition.

A.4 Product Assurance: Proposes IV&V as a transition tool, as training element, to provide support tools, and identify errors in products and data.

A.5 Transition: Problems with quality of data and product at transition; proposes IV&V by the support agency as a transition tool; lack of guidance in support area.

B.4 Capital Investment: Discusses need for commonality/modularity in facilities, tools, and training devices, discusses VHSIC, fault-tolerant hardware, multiprocessor architectures, variable task systems, and their implications.

C.3 Design Attributes: Discusses the lack of and the need for design for testability in both hardware and hardware from an ATE perspective.

C.4 Documentation: Errors/inconsistencies in documentation; proposes standardized data item descriptions; proposes IV&V by support agency as partial solution.

D.1 Skills: Discusses management training needs; lack of and deferred training funds.

D.2 Availability: Shortage of skills; lack or deferred training; recruiting problems, retention problems, manpower requirements baselines.

D.3 Incentives: Discusses attrition problem and some of the causes; lack of adequate career progression.

B.19 Final Report of the Software Acquisition and Development Working Group, July 1980.(SADWG)

Funded by: Assistant Secretary of Defense for Communication, Command, Control and Intelligence.

Objective: "(Determine) the efficiency and cost effectiveness of current software acquisition and development practices within the Intelligence Community, and (ascertain) areas which could benefit from better management control."

Methodology: A working group of nine people representing OSD, DIA, NSA, CIA, Intelligence Community Staff, USA, USN, USAF, and Rome Air Development Center accomplished this investigation from February 1979 - July 1980. Inputs came from: 1. In-depth experiences of working group members in C^3I software acquisition and development. 2. Invited presentations from 20 software development corporations doing business with the Intelligence Community. 3. Studying actual case histories of four software development projects.

Structure: A single report with three sections (Intro, Industry Comments, Case Histories) and three appendices (Summary of Industry Comments, List of Software Development Commandments, Measureable Milestones).

Problem Areas: A.1, A.2, A.3, B.2, C.2, C.3, C.4, D.2

Details of Problems Addressed

A.1 Requirements: "Projects, and subsequently contracts often get started with inadequate planning. Also, because of ambiguous or vague requirements, there is often a lack of understanding between the government and contractor as to what is to be delivered." (pp. 1-4)

A.2 Management: "Software development projects are being conducted with a lack of good management practices (i.e., poorly trained managers, inadequate record keeping, insufficient management tools, and misdirection of emphasis at the various development stages)." (pp. 1-4)

A.3 Acquisition: "There is often a mismatch between the contract type and the complexity of the work to be performed." (pp. 1-4) C^3I systems are unique and affected by a unique set of acquisition and development problems. Factors are:

- Generally one of a kind
- Software dominant
- "Finished requirements do not exist with military software systems because requirements are constantly changing and the systems, as a result, are never complete."
- "C^3I systems generally have a development cycle of more than five years before becoming operational." (pp. 1-6)

B.2 Tools: "Everyone agrees that software development productivity must be increased on the part of both the government and contractors. One method of increasing productivity is through the use of available productivity tools. However, because productivity tools are not well understood and difficult to use, instead of enforcing their use, the government only pays them lip service. At present, there is no easy method to quantify the value of productivity tools. Such a method needs to be developed." (pp. 1-6)

C.2 Software Metrics: "Because the government must estimate system costs often years in advance of the actual procurement, the estimates are usually wrong. And because contractors must bid on systems before they are designed, their estimates are also usually incorrect. Neither the government not contractors have adequate means to estimate lifecycle costs with any reasonable degree of

accuracy. The current state-of-the-art in life cycle cost estimating is grossly inadequate." (pp. 1-5, 1-6)

C.3 Design Attributes: "The government inadvertantly impacts costs and schedules by specifying hardware for a particular development before knowing whether the hardware will meet the processing and performance requirements of the proposed system. It has been shown to be very expensive to "shoehorn" software into minimally acceptable hardware configurations, particularly since hardware is less expensive than software." (pp. 1-4) "Software development is unique. The software industry is the only industry required to build usable products right the first time without benefit of intermediate development stages such as prototyping. Although prototyping is an accepted practice in other less complex industries, it is not in software development. Without prototyping or some sort of intermediate development stage, risk factors are significantly increased." (pp. 1-5)

C.4 Documentation: "There are a multiplicity of (documentation) standards within the government which cause inefficiency and confusion. Currently, standards are not precise enough to eliminate misunderstanding between contractors and the government. The level of detail is often open to interpretation." (pp. 1-4)

D.2 Availability: "Security requirements impact software development costs because work cannot begin on a project until the required personnel have been cleared by the agency for whom the work is to be performed. There are built-in delays because interagency transfers of security billets take an inordinate amount of time, and there are no interagency agreements on standards for the investigation or authorization of security clearances. This impact is rarely considered by the government and contractors in estimating costs and schedules. These costs are difficult to estimate because delays for clearances depend on the individual consideration and the agencies involved." (pp. 1-5)

B.20 Predictive Software Cost Model Study, Volumes I-II, 6/80. (S/W Cost Model)

Funded by: Air Force Wright Aeronautical Laboratories

Objectives: Determine feasibility of software support cost model

Methodology: Review literature, site visits to ALCs

Structure: Detailed Report (2 volumes)

Problem Areas: C.2

Details of Problems Addressed

C.2 Software Metrics:

a. Data Collection - Data collection on software support does not, in general, go to the breadth and depth desirable to support a good, solid model of development effort. (p.55)

b. Data Lack - A lack of definitive historical data on software operation and support and an insufficient understanding of driving factors has hampered support cost estimating and assessment of its contribution to life cycle costs. Bases on which models are built are extremely limited. (p. 133)

B.21 Software Requirements for Embedded Computers, A Preliminary Report, 3/80. (S/W REQ-EC)

Funded by: USAF to Rand Corporation

Objectives: Focus on early acquisition problems regarding software

Methodology: Interviews with program personnel

Structure: Preliminary Report

Problems areas: A.1, A.2, A.3, A.5, C.4, D.1

Details of Problems Addressed

A.1 Requirements: A major issue has been the frequent failure of the first products of software development efforts to meet the user's operational needs. One of the most consistent study results has been the identification of software requirements management as a major problem area. The term "requirement" has yet to be satisfactorily defined; it means different things to different people. (p.1)

a. Requirements Management - The most frequently cited requirements problem is the need to respond to late - appearing changes in the requirements baseline. One of the primary tasks of program management is to ensure that such changes are accurately reflected in the requirements the contractor must satisfy. Requirements management is at the heart of and inextercably tied to the entire software development process. A false assumption is that the software development process can easily accommodate change. (p. 2)

b. System Level Requirements - Recent software requirements research has not focused on the early activities leading to system - level requirements which may already constrain software design flexibility by

specifying architectures. Software, despite its mission - critical nature, is not routinely treated as a major decision variable from the earliest stages of the acquisition process. Many software decisions are still made indirectly. (p.3)

A.2 Management:

a. Software Subcontracting - For two command and control systems studied the software subcontractor did not have an overall product orientation. The software package is often developed by a subcontractor who has little knowledge or experience in the overall application area. With associate contractors none may have total product orientation or much experience with developing pieces of command and control systems let alone complete systems.

b. Management Approach - Project to project differences in overall understanding of tasks, perception of a coherent product by the contractor, contractor experience and concreteness of requirements suggest a need for correspondingly different management approaches.

A.3 Acquisition:

a. Source Selection - Contractor's past experience or demonstrated capability in computer or software technology was not a significant factor in the source selection process. (p. 9)

b. Contractor Pre-RFP Effort - Contractor studies prior to the release of the RFP often lock-in the contractor's direction on the software before the Air Force has addressed the software issues informally or formally. (pp. 10,11,15)

A.5 Transition:

a. Effects of Change - In responses to new or changed requirements, developers tend to leave early decisions intact, fearing the impact of changes on budget and schedule. Instead the changes are made at lower levels, usually by complicating the software. (p. 16)

b. Advanced Technology - Proposal writing teams are usually made up of the contractor's advanced technology personnel, who are biased toward the most advanced technology feasible in system design. This has led to trouble where those responsible for the software don't share the optimism of being able to implement the technology. Often little weight is given to practical issues in preparing the advance technology aspects of the RFP. Where Phase 0 contracts are awarded, this is less of a problem. (p. 21)

C.4 Documentation: The producers and SPO reviewers of software documentation do not know enough about the specific information needs of the intended audiences - e.g., mission support, training. The Air Force is paying more for documentation (in direct and indirect costs) than is warranted by its ultimate utility. Minimum essential information for each group of users should be determined and standards established. (p. 19)

D.1 Skills:

a. Flexibility, - Often software is assumed to be infinitely flexible; however, basic hardware/software trade off issues may be decided in the absence of software expertise. One effect is a reduction of the design and implementation flexibility available during full-scale development. (p. 4)

b. Skill Shortage - Long standing resource problems, the "start from scratch" character of most program offices, the limited transfer of experience from one program to another are all major impediments to lasting improvements to software acquisition management. The primary resource problem is attracting, training, and keeping talented people. Software problems on a number of large programs have been quickly and effectively turned around by the efforts of single individuals. What are the characteristics of such people and how can their skills be accessed? (pp. 17,18)

c. Software Acquisition Expertise - The "kind" of software expertise is as important as the "number" of people required. Air Force programs are not producing the most needed kinds of expertise. Such people need a background in computer science and the engineering applicaiton such as radars, etc. (p. 19)

B.22 Computer Software Contract Administration, 2/80. (S/W ADMIN)

Funded by: USAF/HQ Contract Management Division

Objectives: Establish Software Needs of CMD Personnel

Methodology: In-House Study and Workshop

Structure: Coordinated Study Report

Problem Areas: A.2, A.4, A.5, D.1

Details of Problems Addressed

A.2 Management:

a. AFR 300 and 800 Series Policies - A dichotomy exists in the organizational structure and policies at HQ AFCMD relative to AFR 300 and 800 series regulations. (pp. 3-4)

b. Automated Management Techniques - Verification and validation of these (contractor) management systems through the usual CMSEP (Contractor's Management System Evaluation Program) approach will take a new meaning. (pp. 3-4)

A.4 Product Assurance:

Software Reliability - Software reliability is ill-defined. Work needed to be done for proper inclusion in policy. (pp. 4-14)

A.5 Transition:

Firmware Management - Guidance needed in following areas: content and structure of deliverable documentation; subcontracting provisions; data rights; effects on logistic support capabilities. Firmware management disciplines should be tailorable. (pp. 3-9)

D.1 Skills: Must define necessary skills for management and technical personnel. Feasibility of standard position descriptions for engineering and quality assurance personnel (for ECRs). (pp. 4-16)

Embedded Computer Resource Training - Presently no AFCMD training program exists which identifies the responsibilities of the functional divisions. The need is obvious. Comprehensive training course with short term and long term objectives is required. (pp. 2-15)

B.23 Final Report of the Joint Logistice Commanders Software Workshop, 10/79. (S/W WORK-79)

Funded by: Joint Logistics Commanders

Objectives: A plan for implementing joint policy recommendation

Methodology: Workshop panels - industry and government

Structure: Report

Problem Areas: A.2, A.3, B.2, C.2, C.4

Details of Problems Addressed

A.2 Management:

Management Discipline - Project managers have no measures of progress or assurance that the software will perform at an acceptable level. (pp. 2-5)

A.3 Acquisition:

General Policy - No general policy for common acquisition framework for joint services. (pp. 1-2)

B.2 Tools: All services appear deficient in the procurement of the necessary tools and documentation. (pp. 1-2)

C.2 Software Metrics:

a. Quality Assurance - Difficulties in QA are the lack of well-defined, consistent requirements, differences in SQA approaches, and inavailability of experienced personnel. (pp. 2-4)

b. Acceptance Criteria - There is a lack of recognized acceptance criteria, a lack of DoD standardization, and a lack of historical data on which to base acceptance criteria and procedures. (pp. 2-5)

c. Error Data - The paucity of software error data for joint service programs makes it impossible to develop an accurate error model to help predict software reliability. (pp. 2-5)

C.4 Documentation: There are a number of diverse standards related to software documentation within the services. There is no defined methodology for establishing minimum documentation requirements. (pp. 2-3)

a. Standards - Partial and inadequate standards coupled with a lack of standardization of Data Item Descriptions lead to delivered software products which are often unsatisfactory. (p. 2)

b. Terminology - Terminology and definitions vary among the services even though the subject matter is identical. (pp. 2-3)

B.24 Proceedings of the Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resource Management, 8/79. (MGMT-79)

Funded by: Joint Logistics Commanders

Objectives: Establish triservice common software policy

Methodology: Triservice and industry working groups

Structure: Unofficial Report

Problem Areas: A.3, A.4, A.5, B.2, C.3, C.4

Details of Problems Addressed

A.3 Acquisition:

a. Systems Level Issues - The general assessment was that system level issues in the final analysis may have a greater impact on software success than the software acquisition issue. (p. 15)

b. Triservice S/W Acquisition Policy - Such a policy is essential. (p. 16)

c. Weapon System Acquisition Policy - Deemed inadequate regarding the embedded software elements part of a larger system. (p. 17)

d. Post Deployment Support - Should be addressed early in acquisition prior to FSED. (p. 17)

A.4 Product Assurance:

a. Operational Test - Extremely difficult to test ECS in close to real environment, hence many systems reach deployment without true operational effectiveness being measured. (p. 16)

b. IV & V - No agreed upon definition or how it is to be contracted. (p. 16)

c. Quality Assurance - The need and degree of quality assurance for software on the DoD policy level is described. In addition to policy a guidebook and handbook are required. (pp. 127-139)

d. Software Acceptance - The problem is stated as "the acquisition program manager often has no assurance that computer software, if accepted, will perform operationally at an acceptable level." The needs for acceptance criteria and a guidebook are expressed. (p. 161)

A.5 Transition:

a. Firmware - More rigor, especially during test, should be employed because of the inflexibility of the medium and multicopy production. (p. 15)

b. Microprocessor/Microcomputers - Very deeply embedded micros precludes individual test and qualification. (p. 15)

B.2 Tools

PASCAL - PASCAL is becoming de facto industry standard and not in DODI 5000. 31.

C.3 Design Attributes:

Preliminary Design - Better statement needed of level of detail at PDR for CPCIs and corresponding documentation. (p. 17)

C.4 Documentation: There are diverse sets of documents and documentation requirements. A single set (subsetable) should be defined. (pp. 93-97)

B.25 Computer Technology Forecast and Weapon System Impact Study, Vol. I, II, III, 12/78. (COMTEC-2000)

Funded by: USAF

Objectives: Project future computer resource technology applications

Methodology: Conference and panels briefed by experts

Structure: Application oriented reports (Approved)

Problem Areas: A.1, A.2, A.4, A.5, B.2, C.3, D.1, D.2

Details of Problems Addressed

A.1 Requirements:

Complexity - For aeronautical systems software, implementation of future functions will become increasingly complex and more expensive to engineer. (pp. 2-6)

A.2 Management:

Standardization - Standardization will have to occur so as not to isolate DOD from commercial market technology advances. Computer architecture standardization has the potential of divorcing DOD from commercial support software and mass produced microcomputers. (pp. 203)

A.4 Product Assurance:

Testing - Dynamic testing is not adequate to produce the degree of reliability required in many systems. As systems become more complex, the inadequacy will become more apparent. (pp. 2-3)

A.5 Transition:

Rapid Reprogramming - For aircraft sensors a rapid reprogramming requirement exists based on threat, tactical, and doctrinal changes. (pp. 5-44)

B.2 Tools:

a. Support Software Sharing - Current system applications are constrained by the lack of a viable networking system to enable support, operational, and development commands to share support software. (pp. 2-6)

b. Test Tools - Software test tools have not been combined to produce an effective programming environment. (pp. 4-39)

c. Resource Estimation - Resource estimation is needed in the development and support of software systems. These include manpower, schedule, sizing, timing, complexity, and cost. Data collection is necessary. (pp. 5-79)

C.3 Design:

a. Design Techniques - For Aircraft/Weapon interfaces design techniques are needed that will allow mapping from system requirements to an effective flexible system design. (pp. 2-5)

b. Advanced Sensor - In software, the development of appropriate near-real-time algorithms and data representations for sensor processing is likely to be both a major schedule and cost factor. (pp. 2-6)

c. Distributed Processing Needs - The entire area of distributed processing needs to be better defined and better understood. Included are needs for system level technology, modeling, security, distributed data bases, fault tolerance, design methods, and analysis methods. (Vol, III, pp. 1-19)

d. Distributed Systems - Tools for accomplishing hardware/software/firmware trade-offs of distributed systems are not yet adequate. Limitations in the efficient use of distributed systems will arise from software constraints. Adequate analysis tools, methods

of problem partitioning, and techniques for controlling software/hardware modules are needed. Modelling and development tools do not exist. (pp. 4-29)

e. Distributed Executives - Distributed systems make clear the need for distributed executives. (pp. 5-37)

D.1 Skills:

Software Engineering - Software engineering is not a science and there are few accepted methodologies and tools available to support the necessary engineering functions. The process is labor intensive. (pp. 5-78)

D.2 Availability:

a. Productivity - Present (software engineering) productivity is too low and growing too slowly to keep pace with rapidly growing hardware capabilities. The demand for software personnel will exceed the supply by 30% by 1985. (pp. 4-41)

b. Labor Intensive - Software is extremely labor intensive; as a result software is rapidly becoming the limiting factor in large systems. (pp. 4-41)

c. Personnel Shortfalls - This panel report focuses on existing shortfalls in 511XX and projected shortfall in 51XX career field skill areas. Civilian personnel are also short in numbers. Future needs are projected to be far more serious. Personnel limit the capability of computer systems. The power and flexibility of computer systems cannot be utilized without the proper numbers and skills of software specialists. Since computer systems form an integral part of today's management and weapon systems, software personnel shortfalls will keep the Air Force from fulfilling its operational mission. (Vol. III, pp. 1-5)

B.26 Operational Software Management and Development for U.S. Air Force Computer Systems, 1977. (OPER S/W M&D)

Funded by: Air Force Studies Board

Objectives: Improve effectiveness in operational software

Methodology: Summer Study

Structure: Final Report

Problem Areas: A.2, A.3, A.5, B.2, C.3, D.1, D.3

Details of Problems Addressed

A.2 Management:

Management Discipline - Software is not identical to hardware except in the need for discipline. Lack of a sure way to assess progress furthers the problem. Each project has unique problems and challenges requiring a unique set of specific management techniques. (p. 20)

A.3 Acquisition:

a. Requirements - Contributing to high risk in cost and schedule is the general incompleteness of the requirements - particularly regarding anomalous and extreme conditions, interfaces with other systems, and collateral functions. Realism in requirements is also a problem. It may be difficult to resolve problems of realism early. Life cycle aspects and total systems aspects must bear on software requirements. (pp. 15-19)

b. Procurement - Since the development contract is generally consumated before system development is sufficiently understood, it follows that lowest price as a criterion for contract award will get too little effort, an improperly short schedule, inadequate staffing, and possibly insufficient facilities. Changes of scope can be used to make up deficiencies, but a job which starts off

badly has little chance of becoming right through the change of scope process. Especially with a fixed price bid, this is a delicate problem. The Air Force lacks good ways of rejecting underbids.

Furthermore, "one shot" or "turn key" developments imply poor life cycle costs, poor performance, unresponsiveness to changing requirements and a hardware first, software catch up philosophy. (p. 22)

A.5 Transition:

User Maintainer Participation - Low or non-participation of user/maintainer organizations hampers accurate requirements definition implying poor transition coordination and poor preparation for post delivery. (p. 21)

B.2 Tools: Problems cited with tools include proliferation of different tools for similar purposes, a question of who and how should tools be funded, and issues of transportability and reusability including diversity of development computer types, contractors' own tools, and GFE risks. (p. 59)

C.3 Design Attributes: Hardware decisions are generally made before software development begins, and the software (because it is so "flexible") is left to fix any problems. "Flexibility" also means complexity, development difficulty, and use/maintenance problems. Trying to fit a job into an under-sized and under-powered computer is a sure way to high cost and risk of failure. (p. 19)

D.1 Skills: Lack of strong systems engineering capability leads to inflexible emphasis on cost, schedule, component items, and policy at the risk of the total system. The tour of duty (particularly of junior officers familiar with system details) is not job related. Software management expertise is not widely available. Courses are needed and case history information is lacking. A major need is in mature knowledge of how computers and systems work from a systems engineering context. (pp. 20-21, 65)

D.3 Incentives:

Career Concept - Career path and career concept in software engineering is not clear. Management requires technical understanding but more from a systems engineering sense than in a how-to-program sense. (pp. 67-68)

APPENDIX C

Case Studies

INTRODUCTION

Appendix C contains case studies of actual problems encountered in developing and supporting software for weapon systems. The problems are described within the context of the weapon system, and the problem and its data are used to derive a conclusion. Where possible, actual numbers are used to give an idea of the scale of the problem. Appendix C should be read as examples of the real-life problems of software development and not as indictments of the projects described.

C.1 Fleet Combat Direction Software Support Activity (FCDSSA) Resource Shortage

An in-depth functional analysis of the FCDSSASD (Fleet Combat Direction Software Support Activity in San Diego) was performed as part of the SEATECS (Software Engineering Automation for Tactical Embedded Computer Systems) program managed by the Naval Ocean Systems Center (NOSC). This functional analysis shows some of the problems and dynamics faced by Navy software production groups today. The following observations are derived from the final report which was prepared by Systems Consultants, Inc., October 15, 1980.

The report projects serious shortfall in programmers and computer support resources in the next decade when planned budgets are compared to projected workload. The problems addressed relate to taxonomy areas B.4 (inadequate capital for the process and restricted host facilities) and D.2 (shortage of competent developers).

C.1.1 Background: The functional analysis of the FCDSSASD was initiated because of NAVSEA concern about the adequacy of the FCDSSASD resources, existing and planned, to meet future mission requirements. FCDSSASD along with a similar organization in Dam Neck, Virginia (FCDSSADN) provide in-service support for the Navy Tactical Data System (NTDS) and other embedded systems aboard surface ships. In 1980, FCDSSASD was supporting multiple systems totalling more than 1.5 million lines of source code. By 1990, the supported code is projected to double. The O&MN (Operations & Maintenance Navy) funds allocated to the command for software maintenance is projected at a constant level (adjusted for inflation) despite the projected growth in workload.

In addition, FCDSSASD has a fixed allocation of space suitable for equipment installation. This allocation reached saturation when a third host support system was installed in 1979. With the projected increase in demand on FCDSSASD, the requirement for support equipment and the corresponding space is expected to increase.

The military billets and civil service ceilings allocated to FCDSSASD are projected at current levels. The estimated workload increase might be accommodated by expanding the contractor workforce. This will be difficult given the current and worsening shortage of area of embedded computer system development and maintenance. In addition, the increased workload will seriously stress the ability of FCDSSASD personnel to perform operational and technical tasking, as an increasing proportion of the staff will be required in management and support functions.

C.1.2 Analysis: Figure C.1-1 shows the labor resources required to meet the projected workload compared to the available resource. By 1985, critical resource shortfalls will occur. By 1990, the shortfalls will be catastrophic, in that over 600 additional people over that available to meet the workload demand will be required. These projections are based on a conservative estimate of FCDSSASD workload and industry studies which predict a 5% annual growth rate in the population of data processing professionals through 1990.

Figure C.1-2 shows the projected ADP equipment resource requirements against availability. Without significant capital investment in ADP equipment for system test support and software development support, no amount of labor resource will be able to meet the projected workload. By 1990, the estimated annual shortfall in the test facilities "mock-up" hours will be over 190,000 hours.

C.1.3 Impact: The FCDSSASD analysis is one example of a major problem facing DoD in mission-critical software. Accelerating growth in the workload will soon overwhelm the production resources and thereby threaten the DoD mission, for software is critical to most modern weapon systems. Given the problem, three courses of action are available:

- Reduce the workload - this is not feasible unless our adversaries cease the introduction of new threats.

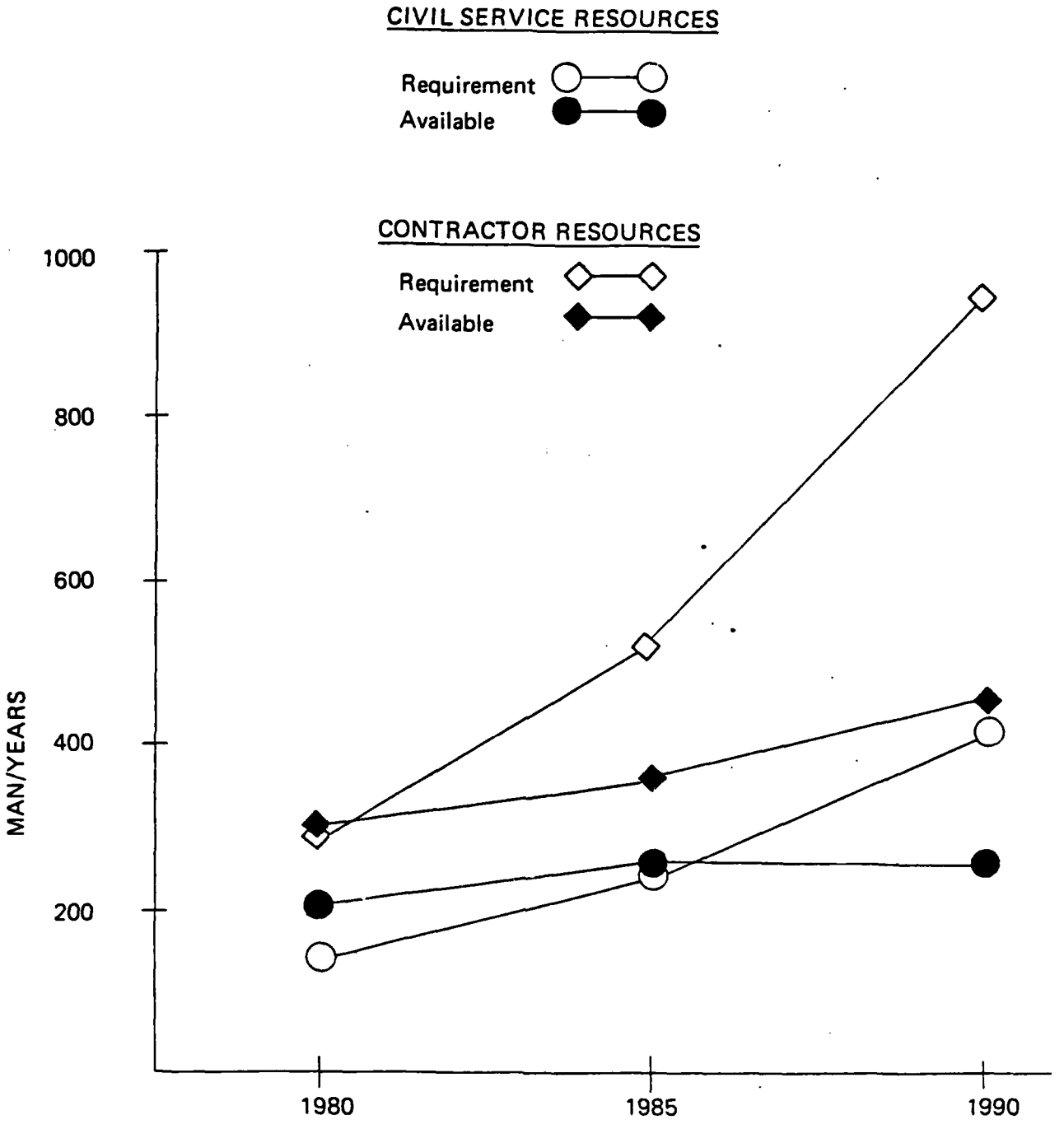


Figure C.1-1 Labor Resources

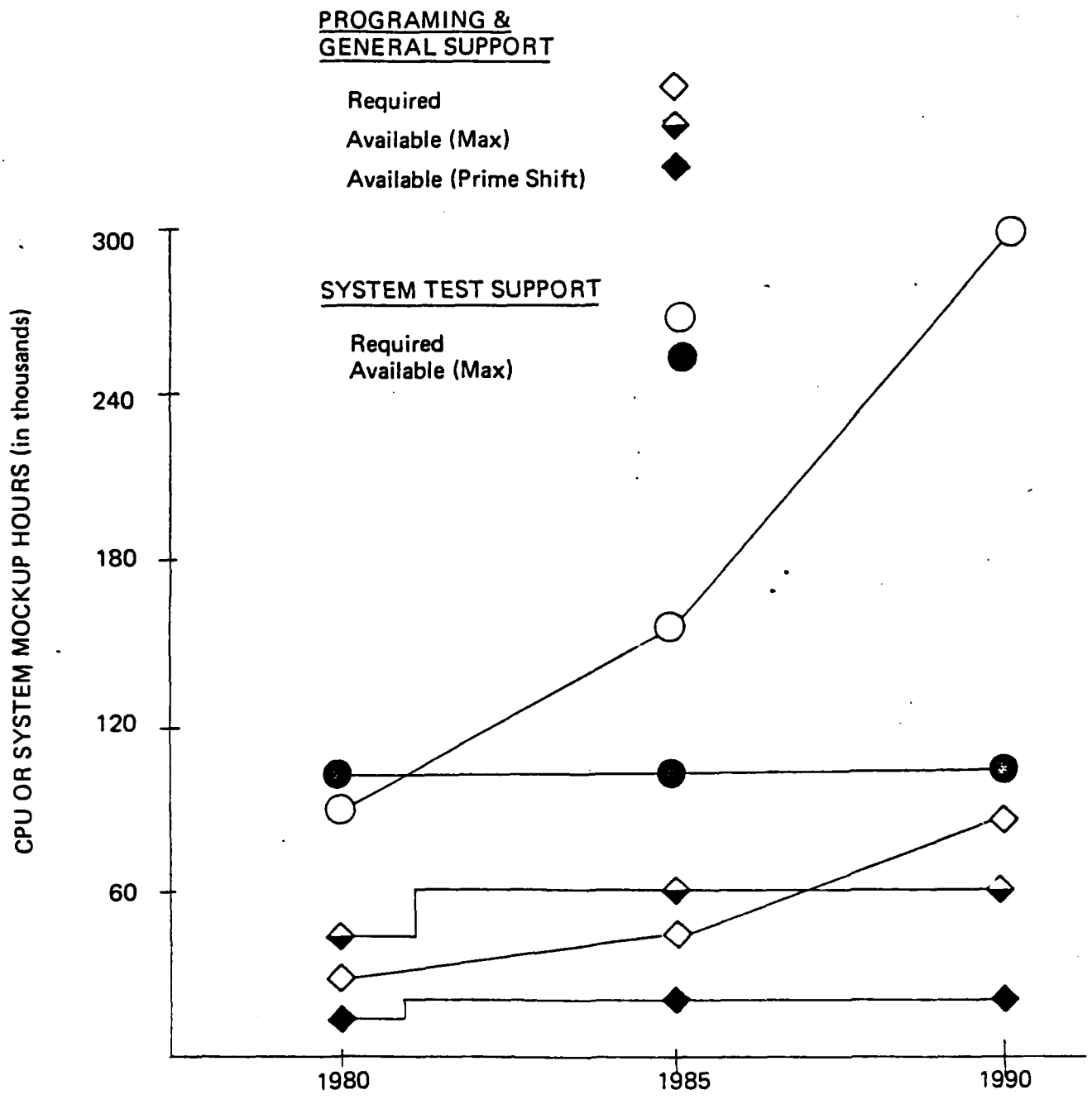


Figure C.1-2 ADP Equipment Resources

- Increase the resources - this is not feasible given the shortage of qualified software professionals. Even if ample resources were available, the cost associated with the increases would overburden the budget.

- Increase productivity - this is a possible solution, but it will involve significant capital investment and it will take several years before any results are evident.

If no positive action is taken, a likely scenario is that timely introduction of new weapon systems and enhancements will gradually deteriorate with a corresponding decrease in reliability as corners are cut. The end result will be inferior DoD military capability and failure-prone systems.

C.2 Opportunities in Reusable Software Components as Exemplified by the Navy Command and Control System (NCCS)

A common problem in software development today is the inability to make use of functionally similar software developed for other systems (see taxonomy area B.3, Reinvention). The reasons for this are: 1) it is difficult to determine if functionally similar software exists; 2) the large majority of software is not designed with reuse in mind; 3) to be reusable, software must be designed and developed according to uniform and rigorous software engineering practices; and 4) high-level policy direction mandating reusability is missing.

The Navy Command and Control System (NCCS) Ashore is an example of the opportunity in reusable software. It consists of four major computer systems, each developed independently from the mid 60's to the mid 70's. The following discussion and analysis is not to imply that reusable software should have been used, for the state-of-the-art at the time of development would not allow this, however, the opportunity cost shown is real and represents but one small piece of the opportunity cost in reusable software.

C.2.1 Background:

NCCS Ashore consists of four computer systems developed for unique and independent use originally, but integrated into a command and control communications network to form NCCS around 1975. Although independently developed at different times, each of the system's software exhibits a high degree of common function as described below:

- Communications Processing: All four systems have a common communications processing function responsible for terminating the various Navy and DoD communications lines and handling their protocols. Individual line handlers are embedded in an overall communications architecture and each is present at those nodes where they are required.

- Message Processing: As with communications processing, message processing is a common function. The function provides parsing and error processing as well as reformatting and buffering between the inner system and the communications processing function.

- Ocean Surveillance, Blue Force Tracking, Status of Forces Data Base Maintenance: All these functional areas, where required at individual nodes, are performed by similar software.

- Command Support: The area of command support entails reporting, command and analysis displays, analysis activities, data base queries, etc. This area is probably the most variable of the functional areas, yet substantial commonality exists in the display, report formatting, and analysis support areas.

Each of the four systems is supported independently. Since the size of the software subsystem in each system ranges from 100,000 lines of source code to over 1,000,000 lines of source code, the opportunity cost in redundant support of functionally common elements is potentially very large. Whether or not it would make sense to attempt to capitalize on this opportunity at this point is unclear due to the significant redevelopment cost that would be entailed.

C.2.2 Analysis: An estimate was made of the percentage of each of the four software systems that represents unique function. The remainder represents common function. This was then compared to the support budgets since 1976 to estimate the opportunity cost (potential savings) if common software components, supported by a central group, had been used from that time. Figure C.2-1 shows the result of this analysis. Since 1976, approximately $34 million has been spent on support (maintenance) and enhancements of the four NCCS Ashore software systems. If we assume that cost is equally distributed over all functions, then approximately $16.6 million went to unique function software and $17.4 million to common function software.

If the common function software had been implemented in common software components such that a single support group could maintain and enhance it, it is estimated that $10 million of the $17.4 million dollars is representative of the potential cost savings. The original opportunity cost related to reusable software is larger than this because there would be a saving in the original development.

	OBS (11/77)	FHLT (1974)		ASWOC (1970)		NWSS (1974)		TOTAL MAINTENANCE	
YEAR	TOTAL	a. TOTAL	b. UNIQUE	a. TOTAL	b. UNIQUE	a. TOTAL	b. UNIQUE	INCURRED 1+2a+3a+4a	MODEL 1+2b+3b+4b
1976		1.2	.3	.598	.179	2.2	1.1	3.998	1.579
1977		1.2	.3	.493	.148	2.2	1.1	3.893	1.548
1978	.8*	1.2	.3	.321	.096	2.2	1.1	4.521	2.296
1979	.8	1.5*	.375*	.518	.155	2.2***	1.1	5.018	2.43
1980	1.6	1.5	.375	.311	.093	2.141	1.071	5.552	3.139
1981	1.6	1.5	.375	.556**	.099	2.335	1.168	5.991	3.242
1982	.8	1.5	.375	.617**	.125	2.127	1.064	5.044	2.364
TOTAL	5.6	9.6	2.4	3.4	.9	15.403	7.703	34.017	16.598

Figure C.2-1: NCCS MAINTENANCE AND ENHANCEMENT COST CALCULATIONS
(COSTS IN MILLIONS)

* Earliest year for which figures were provided. Any prior cost figures are extrapolations to the year when placed in maintenance, if known.

** Support S/W costs of $225K in '81 and $200K in '82 are included in Column a but have been deducted before applying commonality factor.

*** This and prior years are estimates.

C.2.3 Impact: The obvious impact of the prevalence of reinvention is high cost of software. It is reasonable to assume (based on the NCCS example and the ones outlined in the Notes below) that the opportunity cost of reusable software components in DoD is in the hundreds of millions of dollars. This cost in itself is reason enough to vigorously pursue reusable software components, but there is another perhaps more important reason. The shortage of qualified personnel tied to the increased use of software in weapons systems is creating a situation where development is being squeezed by the support of existing systems. The introduction of reusable components can significantly relieve the resource demands thus assuring continued responsiveness to new threats through the introduction of new or enhanced weapon systems.

Note 1: Some would argue that DoD systems are much too complex or unique to benefit from reusable software. But even a short study of DoD software will reveal abundant opportunities for reusable components. The number of real-time executives, file management subsystems, display software, report generators, navigation routines, etc., that have been developed independently but which represent very similar functional capability is large. Some of the most complex software systems developed by DoD are simulation/stimulation systems used for test and training. The redundant software in these systems alone measures in the tens of millions of dollars.

Note 2: The Navy has begun a program based on reusable components called Restructured NTDS. It is estimated that over the period 1986 to 1990, $20 million could be saved on trouble reports and small engineering change proposals to NTDS if this architecture were used for DD-963, CGN-38 and CG-26 class platforms alone.

Note 3: The current procurement practices of DoD inadvertently foster reinvention. One example of this is a situation where three weapon systems were procured from three different contractors. Each system had a common navigation component, in fact, they used identical algorithms. Yet the navigation software was implemented uniquely by each of the three contractors. It is estimated that this redundancy in procurement of software cost DoD over $10 million.

C.3 The Interrelation of Software and System Problems

When analyzing a weapon system development project that is in trouble, it is usually very difficult to separate the software problems from the system problems. The problems are interrelated and no one problem or other difficulty can usually be identified as the reason for a cost overrun.

An example of a surveillance system that had significant schedule delays and cost overruns is described in the following sections. There were many problems encountered including:

- System design decisions that complicated software development (taxonomy area C.3)
- Inexperienced DoD project management (D.1)
- Unstable and inadequate GFE and GFI (B.4, C.3)
- Hardware constraints on software (C.3)

It is difficult to allocate specific cost or schedule overrun to each of these problems because they are interrelated.

C.3.1

Background: In the mid 1970's development was begun on a complex surveillance system. It was to be deployed in the late 1970's after integration with two other surveillance systems being independently developed by other contractors. All three surveillance systems required a special signal processing computer. The original development budget for the surveillance system was around $20 million. The final cost was around $30 million and this was accompanied by a schedule delay of over three years. This delay impacted the integration with the other two systems with additional significant cost impact. Some of the major events leading to this situation were:

- An early system design decision was made by the prime contractor to build a special purpose signal processor computer. When, due to this computer, the development of the system failed one of the DSARC milestones, the computer was abandoned and the milestone repeated using a GFE computer. This resulted in a schedule loss of several years.

- The change in the computer caused the software to be redesigned.

- The GFE computer and associated GFI support software (real-time executive and compiler) were not complete and were still unstable, complicating software development.

- The memory and mass storage of the GFE computer proved insufficient for the software further complicating the software development and requiring upgrades to the hardware.

C.3.2 Analysis: The combined results of the above events put the project under severe schedule pressure due to its potential impact on two other projects. This resulted in further cost growth as the software development was accelerated. The final result was an overrun of $9-11 million in cost and over three years in schedule. Of the $9-11 million, $2-3 million is attributed to problems with GFI software and cost growth in the software development. Given the inter relation of problems it might be argued that more or less cost should be attributed to the software.

Given this example, one is lead to question the computerrelated experience and skills of the DoD and prime contractor management in this project. Several questions might be relevant, such as:

- If the signal processor was key to all three surveillance systems, why was each allowed to choose his own course?

- Why was no cost and schedule reserve set aside to accommodate the known immaturity of the GFE and GFI?

- Why weren't analyses performed in the design phase that would give advance warning on the memory constraints?

The experience of the management of such a project is crucial to its success. Most management of such systems are ill-equipped to deal with technical issues of the type raised above.

C.3.3 Impact: The cost impact for the single surveillance system described was significant. It is likely that the total cost impact to all three systems was much greater. Yet the most significant fact was that a surveillance system, key to our defense, was delayed over three years due to interrelated computer and software development problems.

C.4 Artillery Fire Direction System

The requirements for a computer based tactical fire direction system for support of U. S. Army Field Artillery units was initially set forth in 1959, but development was not initiated until December 1967 and full scale production not released until November 1978. The recorded history of this project discusses some of the problems encountered in the software development for this system over this long period of time.

Problem areas include: changing requirements, taxonomy area A.1; inaccurate estimates of software size and schedule resulting in a lack of adequate personnel assigned to the project, taxonomy area C.2; lack of qualified software development and management personnel, taxonomy reference D.1, D.2, D.3; management problems due to split responsibility for software and hardware development, taxonomy reference A.2.

C.4.1 Background: The Artillery Fire Support program was subjected to a number of changes in user requirements during its development period, particularly increases in requirements for the battalion system in data base size and ammunition type. A major management problem for the project manager during most of the program, was the split in responsibility for software and hardware development. One Army Command had the software responsibility and the project manager assigned to a second Army Command had the hardware and systems responsibility. This split responsibility resulted in imposing differences in documentation standards and instructions on the contractor. This split responsibility was finally resolved in November 1976 by giving the project manager full responsibility for both software, system and hardware.

There was a lack of appreciation by both the contractor and Government for the time required to develop the software, inaccurate estimates of software size which led to a gross underestimation of memory requirements. These underestimates due to lack of experienced software and management personnel with software background resulted in inadequate resources for the software development.

C.4.2 Analysis: The changes in requirements for the battalion system required the addition of random access drum memory units as well as major changes in software requirements increasing costs and causing delays in the schedule.

The underestimates in software size and development effort and the accompanying lack of resources resulted in a stretch out of the contract of over 30 months.

The split in management responsibility for software and hardware development, resulted in conflicting management control over the contractor, causing both hardware and software deficiencies which needed correction and caused further slippages in the program.

Another systemic aspect to this problem was the gradual understanding that the equipment under contract was not an isolated system unto itself, but is part of a larger functional Artillery Support System consisting of other equipments as well. The conclusion here is that during the last decade the Army's concept of automated field systems has become less equipment oriented and more oriented toward entire Battlefield Functional Areas. This has given birth to the challenge to define the range and limitations of distributed processing. There is a need to define this from an operational and economic point of view, as well as from a technical point of view. From the policy view there is a need to recognize that policy aimed heretofore at the management of specific contracted developments must be broadened to view those developments as part of the whole fielded Army system.

C.4.3 Impact: The impact of the changing requirements, the lack of resources due to underestimates and the split responsibilities for software and hardware resulted in extensions of the schedule for the program by over 65 months beyond the original schedule. This also resulted in increased costs.

The use of incentive fees for the contractor on EPR's in the later stages of the contract resulted in improved performances on the program and the final successful completion of the program with production release in November 1978 and initial fielding in 1979.

C.5 Tactical Message Switching Systems.

C.5.1 Background: This example includes data points from two Army managed switching procurements. Both are mobile equipments intended for primary use in the tactical environment and in the interconnection of that environment to the Defense Communication System (DCS) operated in part under the direction of the Defense Communications Agency (DCA). Both systems provide the capability to handle a variety of communication modes, codes, formats, and transmission speeds suitable for interoperating with old and forecasted terminal equipments and for interoperating, as a terminal, with the AUTODIN system. Both systems also provide message service in accordance with ACP 127 procedures which are message formats and procedures negotiated with NATO and other allies. The success of these systems depends upon how well the software and hardware support offer fielding can keep abreast of policy changes within the DCS, policy changes within the allied arena via treaty, and the insertion of new equipment into the system as a result of wearout and/or unavailability of hardware component parts for replacement. In the case of both systems the functionality changes due to policy have been supported by software implemented design changes. Hardware changes have been the result of equipment replacement or attempts to decrease hardware constraints on growing software implemented functionality enchancements.

C.5.2 Analysis: The two systems will be referred to here as System A and System B. System A was built first utilizing commercial grade ADP equipment and inventory modems and cryptographic equipments. System B was developed later utilizing militarized hardware and new parallel developed cryptographic equipments. Since System B was developed in parallel with the field operation of System A some systemic problem generalizations can be made.

System A Analysis

System A began field service in 1971 and was developed under a contract signed in 1968. This was possible since the contract required the use of readily available commercial grade equipments to be transportable via six five ton military vehicles. Needless to say this equipment assemblage was transportable only and not

tactically rugged. The software was the majority of the development effort and software design capability for the message switching functional area was organic to the prime contractor.

System A Software Change Experience

To measure the scope of change to System A an operating period beginning July 1976 through October 1978 was studied. This period was well after the initial fielding date of March 1971 and does not therefore represent initial fielding changes that occur with the fielding of new equipment. Rather, the study period represents the steady state condition typical of communication message switching systems. During the study period 15,908 records (instructions) were modified, added, or deleted in the support and fielded maintenance tools alone. This equates to a 13% modification of maintenance software.

In terms of the software programs necessary to send and receive messsages approximately 14,500 records were modified, added, or deleted representing a 14% modification.

Most of these software changes were related directly to policy changes emanating from DCA in response to the Joint Chiefs of Staff. The conclusion is that it is normal for open loop policy dependent systems to change with a functionally expanding global system. As we look to the future and other expanding forms of distributed processing in communications and other C^2 areas it is clear that functional changes made to our automated equipment will ripple across all associated automated equipments in ways which will appear unscheduled and misunderstood. Since the functionality of modern communications systems and equipments has become implemented more and more within software it is clear that system configuration management must now be defined in terms of functionality in lieu of hardware components and piece parts. There is a need to develop an expansion of configuration management to include a view of collections of systems in terms of common functionality representation, taxonomy reference A.1, changing requirements and A.2, confirguration management.

System A Hardware Change

The basic driver for System A hardware changes was the initial commitment to commercial grade ADP equipments. Within the field 24 hour per day operational environment it was placed, System A required a regular ongoing hardware change policy which was named "piecewise modernization." This policy required that replacement of degrading hardware meet one or both of two criteria: that the subsystem is neither reasonably repairable nor economically maintainable; that the modification be cost effective in terms of maintenance and/or operational costs prior to the planned end of life cycle or 1984. There were many hardware redesign changes made primarily in the power subsystem, the environmental/control subsystem, and the computer subsystem.

Within the computer subsystem the front end computers (three each) were replaced with a more modern minicomputer emulating the original computer selection. This was necessary due to nonavailability of core memory stacks and a variety of piece parts and logic boards. The emulation technique allowed the preservation, and indeed enhancement, of the software implemented functionality, taxonomy reference C.3, design attributes.

The primary conclusion reached from this experience is that the expectation of parts being unavailable over the System A life cycle due to use of commercial equipments created the need for an end of useful life economic analysis of hardware change through which knowledgeable support management decisions were made. The life cycle of a commercial computer asset depends upon the market seen by the manufacturer and not the specific weapon system manager who may choose to use it. In addition, the continual emergence of new electronic hardware technology and its migration to commerical and military products must be taken into account by making the software implemented system functionally immune to hardware technology transfer. This is a primary motivator for the Army MCF program in which dimensions, interfaces and the instruction set architecture are the only hardware parameters held constant over time.

System A Support Personnel Change

The support of System A by government only personnel was accomplished via a transition plan between March 1974 and March 1975. It has become clear in 1973 that the contractor software support of System A was becoming too costly for both the contractor and the Army. It is not feasible to turnkey the support function from one group to another for a complex system such as System A which utilizes five computers and over 150K lines of code in the main switching function programs. Because of this, transition occured in steps. First the direct system engineering functions and management functions were bought "in-house." Next contractor personnel and equipment were placed among assigned civil service engineers. Finally, the contractor personnel were withdrawn. A policy was developed to provide group representation through internal training and planned turnover of personnel over time. A minimum essential level of seven personnel was established below which corporate knowledge could not be maintained over time. The conclusion from this experience is that personnel resources must be managed in terms of system functionality and complexity if workforce regeneration is to be fueled correctly. Documentation alone, however good, is not sufficient to sustain corporate knowledge and understanding over time. Better economic management models based upon functionality and human cognition needs are required, taxonomy reference D.1, D.2, personnel skills and availability.

System B Analysis

As previously stated above the Army experience with System B development overlapped System A field support. Both functions were supported by the same Army organizational unit. System B requirements were more demanding than System A. Beside the requirement to achieve a greater than 200% reduction in equipment volume and weight over System A, the System B functionality was also greater than that of System A. Like System A, the System B development effort was largely software implementation oriented. Unlike System A, a large portion of System B was developed via subcontract to a firm specializing in communication software.

System B Software Change Experience

Based, in part, upon what had been learned with System A, the functionality requirements imposed by the evolving DCS AUTODIN system were withheld from the system B contractor so as to achieve and hold a contract technical baseline. This strategy permitted an integrated hardware and software solution to emerge so modifications could be contractually made without loss of management control. In addition, after successful CDT and DT/OTE testing two development models were provided to US Army Europe for early fielding so as to get early field troop experience. Managed stepwise upgrades were then planned and executed. The first limited AUTODIN upgrade consisted of nineteen function changes and was fielded in the spring of 1980. A second extended AUTODIN upgrade added six more functional changes and was available in the fall of 1980. The systems were then used in four major field exercises during which some twenty-five specific problems were detected and resolved in four more software releases, the last being in the winter of 1982. In addition, this release included five enhancements generated as the result of the USAEUR experience. After this another release was developed and fielded in the spring of 1982. This change included 17 more specific changes originating from the AUTODIN evolution. Currently another modification is in progress which includes a new interface design and operational enhancement changes. The conclusion here is that in addition to AUTODIN evolution imposed changes the system must absorb changes due to other new terminal and network control equipments being designed in parallel with or logging with System B. Switching systems must always bear the brunt of interoperability changes that are driven by new terminal equipment and network control designs, taxonomy reference A.1 changing requirements and A.2 management.

System B Hardware Changes

Unlike System A, System B was developed using newer available technology. This provided the realization of the physical size and weight reduction mentioned above. However, it was noticed that the prime contractor provided a large number of non-standard part requests during the System B development. Approximately 75% of non-standard parts submitted in the first five years of the contract had become standard by the end of that period. This suggests that standard

parts are no longer as time invariant as they once were. As newer technology is absorbed into military systems the concept of standard meaning common use over a ten to fifteen year period is no longer true. Again the conclusion must be reached that system functionality must be managed in a way that makes it highly immune to hardware technology change. The Army MCF program is designed to accomplish this. Again what is needed is a new definition of configuration management based upon functionality specification across systems. This configuration management strategy must accommodate both hardware and software solution projections, taxonomy reference A.2, configuration management.

System B Support Personnel Change

Since System B is just now in production with some units in the field on an experimental basis all support functions are completely on contract. However, there is now a group of military and civilian personnel assigned to preparation for the eventual support of System B from an identified Army Software Support Center. This is in conformance with the Army Post Deployment Software Support (PDSS) Concept Plan identified elsewhere in this report. With respect to personnel the implementation requires a 1:1:2 ratio or one military to one civilian to two contractor persons to support several equipments on a shared basis. While this has been determined through Army experience and comparison with other service and industry experience a clear economic management model which includes people, equipment, and software tools and methods must be developed to determine the minimum essential resources required to sustain real support of system/software taxonomy reference D.1, D.2, personnel availability.

C.5.3. Impact: The Army communication switching experience and other experiences not documented here suggests that there are two kinds of change incident on Army communications and C2 system changes which are requirements generated and which provide for interoperations and distributed processing; and changes which occur in random ways due to continual hardware technology improvements and their effect on the market place. The rate of change in both of these change categories is the single systemic cause for military system

software support cost increases. By realizing that operational system functionality must be managed and primarily implemented in a software solution, it is possible to achieve control over system evolution without total redesign. By realizing that management of the logical properties of computers is an extension of this concept of functionality management to include hardware as well as software functional implementation, it is possible to absorb both change generators and to achieve true functional transportability over time. The biggest barrier to this achievement is the lack of economic decision centered management tools and methods to include configuration management tools and methods built upon functionality and the expectation of functionality change, taxonomy reference A.5, rapid technology change and A.2, management.

C.6 USAF F/FB-111 Post Deployment Software Support

It is generally recognized that embedded computer system (ECS) software provides the potential for flexibility in responding to system problems and mission requirements over the life of the system. One example of this flexibility is presented here for the F/FB-111 aircraft. This data comes from ECS Software Management and Support After System Deployment, May 1977, and Predictive Software Cost Model Study, June 1980.

C.6.1 Background: The F/FB-111 fleet includes some aircraft models which contain only analog avionic and other models which are digital. The FB-111A & F-111F digital aircraft contain operational flight programs (OFP) which were developed by Autonetics in 19 for use in the IBM CP-2 embedded computers. Each aircraft contains two identical computers, one for weapons delivery and one for navigation. The CP-2 is a 16-bit word size machine with 16K of memory. Memory fill for the last several years in each case has been 99 percent. AFLC presently has a modification program to replace these computers (on the F-111D/F and FB-111A) with a machine having a newer architecture (MIL-STD-1750A), 64K of memory, 462 KOPS (three fold increase in speed) and 3500 hours MTBF hardware reliability. The program runs through 1987, including recoding the OFP's in JOVIAL J73 language.

C.6.2 Analysis: Resources - Sacramento Air Logistics Center (SM-ALC) is responsible for post deployment software support (PDSS) of the F/FB-11 Aircraft. They support seven OFP's: one for each of the two CP-2 computers on each of the three aircraft type (F-111D, F-111F, FB-111A) plus one OFP for the navigation computer unit, common to all three aircraft. As of December 1979, they had 90 people, mostly engineers, for this PDSS workload: 80 at SM-ALC (20 government employees, 60 contractor personnel) and 10 off-premise contractors. They occupy about 11,000 square feet of facilities. The engineering lab where this is done (i.e., Avionics Integration Support Facility (AISF)) contains equipment which cost $40 million. The support software in the AISF consists of over 700,000 source lines of code. Flight testing requires 120 flight hours per year: this is based on one block change every eighteen months for each aircraft type; or one block change every six months.

Software Change Activity - For data collected over a ten year period (1970 - 1979), about 1/2 of the changes were corrections (both errors and design deficiencies). About 1/3 were refinements (including deletions and optimizations). Most of the remainder (5-15%) were additions to existing capability. It should be noted that these numbers vary widely by system (e.g., the same report noted the A-7D aircraft change activity had been mostly adding capabilities - 40%). The relationship between number of changes implemented and manhours expended is on the order of 1000 hours per change, with a range of 700 - 1600 hours. These hours are based on an OFP change process which includes a feasibility study phase (change requirements/problem analysis), development phase (preliminary design and development including coding), integration/implementation (of all OFP block change requirements), formal test and evaluation (including user OT&E), and documentation and publication/distribution of the production OFP tapes. These hours include much more than code modification.

C.6.3 Impact: The fundamental difference between software and hardware that permits software modifications to be implemented faster and cheaper is software does not go through a production phase and requires no modification kits. With the exception of documentation, the cost and time for a software change is primarily consumed in developing and testing the prototype, which when complete can be immediately sent to the field for operational use. As an example, the following data is from the F/FB-111 program and shows cost and time for implementing comparable capabilities (additional offset aim points and updated weapon ballistics) through hardware on the F-111A/E (analog) aircraft and through software on the F-111D/F (digital) aircraft:

MOD	HARDWARE	SOFTWARE
(1)	$5.278M/42MO	$0.1M/16MO
(2)	$1.05M/36MO	$0.02M/10MO
(3)	$8.0M/78MO	$0.02M/15MO

The hardware implementations were on the order of 50 times more expensive and took four times longer than the software implementations.

APPENDIX D

Current Initiatives Description

INTRODUCTION

Appendix D contains a description of a selected number of the DoD, Tri-Service, and Service projects aimed at addressing the problems in developing, maintaining and managing embedded computer software. The projects (i.e., initiatives) are divided into two broad categories, one for DoD and TriServices initiatives, and one for the three Services. These descriptions are included to give an idea of the scope of projects aimed at improving DoD's use of embedded computer resources software; they are not meant to be an exhaustive list of DoD initiatives.

List of Initiatives Presented

D.1.1 Ada - The New DoD Standard Language

D.1.1.1 Responsible Activity: Ada Joint Program Office, Office of Deputy Undersecretary of Defense, Research and Advanced Technology (DUSDR&AT)

D.1.1.2 Scope of Activity: The Ada program is a DoD-wide program to establish a common high order language for embedded computer systems, and to provide a programming support environment for software developed under Ada. Although the AJPO provides overall management of the program and has specific responsibility for the common interest, the program is executed by the individual services.

D.1.1.3 Description of Activities: The Ada program extends well beyond simple language standardization and will help control the cost and improve the quality of software by facilitating the application of modern software development practices. The Ada Joint Program Office (AJPO), attached to the DUSDR&AT, is managing the DoD effort to implement, introduce, and provide life-cycle support for Ada. It manages the maturation and evolution of the Ada language and support systems. The AJPO coordinates the development of an Ada Programming Support Environment (APSE), and encourages development of the supporting culture, including management and technical discipline, to assure the DoD a consistent, integrated, programming system which will enhance software portability and afford maximum availability of common tools needed to develop and support defense systems software. There are three major Ada program objectives in support of this purpose, language standardization, introduction and acceptance, and support systems.

Language Standardization

First, Ada must be defined as a consistent, unambiguous standard recognized by the DoD and also by the widest possible community. Recognition of Ada as a standard is a necessary step in the realization of portability for software and people. A difficulty experienced by most other computer languages is the failure to control adherence to the formal definition by implementers, and the resulting proliferation of dialects through subsetting, supersetting and inconsistencies.

In pursuing the acceptance of Ada as a standard outside of the DoD, a certain amount of control must be shared with the standards bodies. This is an advantage in that it will provide a baseline for Ada and protect the language from future whimsical or capricious changes by the DoD or any other body. The AJPO is in the late stages of the American National Standards Insitute (ANSI) canvass process to gain acceptance of Ada as an ANSI standard. Through ANSI X-3, the AJPO is pursuing standardization with the International Standards Organization (ISO) Technical Committee 97, Subcommittee 5.

Introduction and Acceptance

Second, Ada must be introduced and accepted in the DoD as early as possible, consistent with the needs of individual components.

There are a number of projects which could benefit from an early introduction of the language. The momentum of the Ada program has produced a climate ripe for early acceptance. However, the advantages offered by the use of Ada will not be realized unless the programming support environment is also available. Therefore, this objective must balance the need for an early introduction of the language against the risk of a premature introduction. Ada should not be employed on a major DoD program until the Ada Programming Support Environment (APSE) is available to support the needs of that project. The AJPO is responsible for providing current information to DoD program managers who must choose a language for their programs. The AJPO consults with those program managers to ensure that the appropriate support systems are developed. Use of Ada as a Program Design Language (PDL) is being promoted and this strategy has already been adopted by some DoD programs.

Support Systems

Finally, the DoD must ensure the provision of life-cycle support for Ada through the development of a robust Ada Programming Support Environment (APSE) to improve productivity both in development and in continued evolution.

Ada is not simply a new language. By design, it incorporates many of the features needed to support modern programming practices. As such, Ada introduces a new culture which will be fully realized when a sophisticated Ada Programming Support Environment is made available and is widely accepted and used. A robust APSE, complete with advanced development and management tools, will provide the opportunities for substantial improvement in life cycle software management. Although each Service employs a different strategy in the acquisition and management of software, there will be a set of tools which can be shared.

The Navy has been tasked to lead a joint service review team to identify and recommend conventions for DoD-supported APSEs; this is in support of a Memorandum of Agreement between OUSDRE and the Service Assistant Secretaries to work toward a consistent set of APSE interfaces. Contracts will be awarded to develop tools targeted to reside on the Army and Air Force funded developments.

Additional Ada Programming Support Environments are expected to be developed independently by academia and industry. These APSEs will not all be compatible with the chosen conventions. However, tools which are sufficiently powerful may be modified to interface with the APSE. The AJPO will foster development of a highly complete and powerful APSE so that it becomes the leading candidate to evolve as the predominant support system. This should encourage designers of independently developed tools to conform to the chosen conventions.

The AJPO is beginning preliminary investigations which will lead to tailoring of modern programming disciplines, supported by automated tools, to the use of Ada. This activity is expected to increase productivity and improve the quality of software.

A consequence of this objective is a requirement for close cooperation with the industrial sector to encourage acceptance of the language and development of Ada products in the marketplace. Cooperation outside the U.S. has also brought much vital technical input. It has made possible an exchange with our allies and a more viable relationship with the the multinational computer and defense industry.

D.1.2 Computer Software Management Subgroup to the Joint Logistics Commanders Joint Policy Coordinating Group on Computer Resource Management (JLC JPC-CRM-CSM)

D.1.2.1 Responsible Activity: This subgroup consists of a single representative from the Army, Navy, Air Force Systems Command, and Air Force Logistics Command. The present lead office of the CSM is HQ AFLC/LOEC. The present lead office of their parent group (CRM) is HQ Army/DARCOM.

D.1.2.2 Scope of Initiative: To address the problems related to the acquisition and support of embedded computer systems, the Joint Logistics Commanders (JLC) established the Joint Policy Coordinating Group on Computer Resource Management (JPCG-CRM). The JPCG-CRM chartered a subgroup on Computer Software Management (CSM) to serve as a focal point for coordination of activities related to the acquisition of computer software used in support of defense systems.

The mission of the CSM Subgroup is to review policies, procedures, regulations, and standards relating to computer software, and forward specific recommendations to the JPCG-CRM on critical areas related to software acquisition management, including software development, quality, testing, and post-development support. These recommendations should describe and justify specific actions to be taken by JLC or DoD agencies. Furthermore, such recommendations should aid in the improvement and standardization of the software acquisition process with the JLC and DoD communities.

D.1.2.3 Description of Activity:

a. The initial review of current DoD policy and guidance in the area of software management was accomplished by the CSM in 1979. The information reviewed was often conflicting, redundant, or in some cases, lacking. A software workshop was conducted in April 1979 (Monterey I) to review areas where shortcomings existed and to make recommendations for the improvement and standardization of the DoD software acquisition process.

The results of the April 1979 software workshop are documented in two volumes and can be ordered from the Defense Technical Information Center (Volume I - Implementation Plan - ADA 103491, Volume II - Proceedings of the Software Workshop - ADA 103485. See abstracts in Appendix B). Principal findings of the workshop were that:

- Differing policies exist among the Services, producing differences in emphasis and nomenclature with varying interpretations and degrees of implementation.

- Within DoD, there are a number of diverse regulations and standards covering the various aspects of software acquisition. Partial and inadequate standards coupled with a lack of standardization of Data Item Descriptions, lead to delivered software products which are often unsatisfactory.

- Difficulties in the implementation of software quality assurance (SQA) arise from the lack of well-defined consistent requirements, differences in SQA approaches by the Services, and unavailability of experienced personnel.

- Unlike hardware, successful development of software cannot be based simply on passing a definitive series of tests at the end of development. However, there is no standard set of software acceptance criteria within DoD.

Key recommendations in the final report of the workshop were to:

- Develop a general policy framework for the joint Services, to address the entire software life cycle.

- Develop a single unified set of acquisition and development standards for joint Service application.

As a result of these recommendations the JPCG-CRM took upon itself a plan of action. This plan of action included five tasks.

- Generate a plan for software acquisition policy.
- Develop a single, unified set of acquisition standards for tri-Service application.
- Develop a comprehensive set of DIDs, subsets of which would be applicable to any DoD software contract.
- Generate a data item description for a contractor's software quality assurance plan as a tri-Service DID.
- Define and develop software acceptance policy, procedures, and criteria for the acquisition of software in defense systems.

Each task has been initiated. A draft Software Acquisition Policy is currently being reviewed prior to publication. Draft standards for tasks two and three exist, and publication for review is expected in the first half of 1982.

b. In June of 1981, the same group sponsored a second workshop (Monterey II) which also included technical personnel from the Services, other Federal agencies and industry. The purpose of this second workshop was to continue the discussion and to review additional documentation and software management areas for standardization with DoD. Five panels were established for this purpose.

- Data Item Description (DID) study review
- Hardware/software/firmware configuration item selection criteria
- Standardization and accreditation of computer architectures
- Estimating software costs
- Software Reusability

A draft report of the proceedings of this workshop was issued in November of 1981. (ADA 109 441)

D.1.2.4 Index to Problem Areas: A.2, A.3, A.4, A.5, B.2, B.3, B.4, C.2, C.4.

D.1.3 Department of Defense - Software Test and Evaluation Activity

D.1.3.1 Responsible Activity: USDRE

D.1.3.2 Scope of Initiative: The objectives of the initiative are to:

- Develop and promulgate policy for DoD components in the test and evaluation of computer software.
- Stimulate the creation and application of improved tools and techniques for software test and evaluation.
- Support the development of guidelines and criteria for use in software test and evaluation.
- Promote uniform and consistent DoD standards where appropriate in the test and evaluation of software.

The initiative is being directed out of OSD (USDRE), but the study effort is being conducted by Georgia Institute of Technology and Control Data Corporation under contract to DoD. The military services including service headquarters, material commands, program managers, operational test and evaluation agencies, and test organizations are included in the study. A sampling of contractors to the government is being surveyed, but an industry-wide viewpoint is being solicited through the NSIA.

D.1.3.3 Description of Activities: The initiative will be completed in four phases:

- The survey phase is currently underway. The contractors are preparing a draft report that addresses both the state-of-the-art in software test and evaluation technology and the current practices within the military services and industry.
- The second phase will be an analysis of the survey by a panel of experts from DoD, industry and academia.

- The third phase will be an assessment of the need for new and/or revised policy.

- The final phase will be the development and promulgation of policy for DoD components to follow in the test and evaluation of computer software.

D.1.3.4 Index to Problem Areas: A.2, A.3, A.4, B.1, B.2, B.4, C.1, C.4, D.1, D.2, D.3.

D.1.4 DoD 5000.5x Initiative

D.1.4.1 Responsible Activity: Deputy Under Secretary (Research and Advanced Technology)

D.1.4.2 Scope of Initiative: The scope is to establish standards for instruction set architectures for embedded computers within DoD.

D.1.4.3 Description of the Activity: The 5000.5x initiative is a proposed DoD instruction that will establish standard instruction set architectures (ISAs) for DoD and set guidelines for additional ISAs. A requirement of the ISA work is ownership of (or free rights to) the architecture. This ownership is needed to ensure the implementation of the architecture can be procured competitively.

MIL-STD-1862 is an ISA for a 32-bit standard embedded computer; it was developed by the Army. The Air Force is coordinating with the Army on MIL-STD-1862 since the Air Force does not have its own 32-bit ISA standard. MIL-STD-1750 is a 16-bit ISA for avionics use; it was developed by the Air Force.

The Navy has had ECR ISA standards since 1968 for 32-bit machines, and since 1973 and 1975 for a 16-bit ISA. It is currently developing a new series of standard ECRs based on these standard ISAs including the AN/UYK-43 (32-bit ISA), AN/UYK-44 (16-bit ISA), and an enhanced modular signal processing (EMSP) AN/UYS-2 (ISA to be determined based on winning contractor -- data as of 6/82).

Expected benefits of a standard ISA are:

- Reduce ECR software development and maintenance costs through use of a standard set of languages and tools, and through possible reuse of existing software.
- Reduce training costs for ECR programmers through the use of standard languages and tools for a variety of ECR applications.

- Reduce software support costs since programming support and testing support would be for a limited set of ISAs.

- Reduce problems in logistics and hardware maintenance.

D.1.4.4 Index to Problem Areas: A.5, B.1

D.2.1 TECPO - Tactical Embedded Computer Program Office

D.2.1.1 Responsible Activity: Chief of Naval Material (MAT08Y)

D.2.1.2 Scope of Initiative: The "Master Plan for Tactical Embedded Computer Resources" dated April 1982, establishes the Navy's master plan for the development, acquisition and management of embedded computer resources (ECR). This master plan covers all aspects of use, development and support of embedded computers within the Navy. It excludes general purpose computers except as needed for support facilities.

The Master Plan outlines strategy and objectives for ECR within the Navy. The strategy is to continue the Navy's significant effort in standardization. The Navy has been developing standard instruction set architectures (ISA) since 1969 (with the 32-bit AN/UYK-7 and in 1973 for the 16-bit AN/UYK-20). The goal of the Master Plan is to develop, improve and replace all standard ECR in an orderly manner. This goal includes software, personnel and training as well as hardware. Figure D.2-1 summarizes the Master Plan strategy.

The program covered by this master plan is being managed by the Tactical Embedded Computer Program Office (TECPO), within the Naval Material Command (NAVMAT). TECPO develops policy and guidance for the use of ECR in tactical digital systems, assigns Systems Commands (SYSCOMs) for development, provides direction, controls budgets for ECR R&D, and appraises results.

In order to implement the Master Plan Strategy, NAVMAT will:

- Competitively develop and procure the AN/UYK-43, an advanced 32-bit ISA. It must replicate the AN/UYK-7 ISA and be able to use AN/UYK-7 software.

- Competitively develop and procure the AN/UYK-44 reconfigurable microprocessor and microcomputer family, a 16-bit ISA. It must be able to use AN/UYK-20 and AN/UYK-14 software. The AN/UYK-44 will be available as a standard chip set, cards, or fully packaged computer.

Problems \ Solutions	Near Term	Long Term
CP-642 obsolescence	AN/UYK-7, AN/UYK-20 AN/UYK-43, AN/UYK-44	NAVY EMBEDDED COMPUTER PROGRAM
AN/UYK-7 obsolescence	AN/UYK-43	
AN/UYK-20 obsolescence	AN/UYK-44	
Lack of standard airborne computer	AN/AYK-14	
Lack of standard signal processor	AN/UYS-1 (ASP). Develop AN/UYS-2	AN/UYS-2 (EMSP)
Standard disk system obsolescence	Develop high technology mass memory	High technology mass memory system
Proliferation of non standard medium scale displays	Develop standard medium scale modular display subsystem	Standard medium scale modular display subsystem
Programming support deficiencies	Upgrade current Navy standard programming support systems; develop Ada based software engineering environment	Ada compatible standard Software Engineering Environment
Proliferation of Navy programming languages and dialects	Phase out divergent dialects; develop Ada language capability	Implement Ada as single Navy standard programming language
Lack of centralized and comprehensive Navy TECR management	Establish single CNO sponsor for development and life cycle support of all Navy standard embedded computer resources.	

Figure D.2-1: Summary of Master Plan Study

- Competitively develop and procure a standard enhanced signal processor (AN/UYS-2) to succeed the AN/UYS-1.

- Ensure the above new standard ECRs are modular to allow for future advances in technology.

- The existing body of support software will be improved, enhanced, and expanded as needed to support new computers, and will be standardized for common usage.

- Survey support software activities to foster wider use of existing software and to identify gaps for 6.2/6.3 R&D opportunities.

- Develop software tools to improve system designer and programmer productivity and accuracy.

- Stress interoperability in all ECR projects.

- Require the use of standard high-order languages (CMS-2 and SPL/I now and Ada later).

D.2.1.3 Description of Activity: The charter for TECPO is to implement the Master Plan and establish policies for Navy use of ECR. TECPO's emphasis is on the acquisition of ECR including the hardware of embedded computers and related support software, and implementation of ECR policies for the Navy.

Four organizations were created to manage the Navy's ECR activities and to ensure the ECR standards are implemented:

- MAT 08Y, the TECPO, functions as principal development activity in the area of research, development, and acquisition of ECR, support software, and related digital equipment.

- PMS-408, in the Naval Sea Systems Command, is the Navy shipboard ECR project office. It is the development activity primarily for shipboard ECR and associated software support.

- AIR-543, in the Naval Air Systems Command, is the Navy air ECR project office. It is responsible for all development activity for ECR for airborne use.
- ELEX-814, of the Naval Electronics Systems Command, is the Navy electronic group charged with the central management of Navy C^3I and computer technology (exploratory development only) R&D programs.

D.2.1.4 Index to Problem Areas: A.2, A.5, B.1, B.2, B.3, B.4

D.2.2 SEEWG - Software Engineering Environment Working Group

D.2.2.1 Responsible Activity: Chief of Naval Material

D.2.2.2 Scope of Initiative: The Software Engineering Environment Working Group (SEEWG) was established by the Chief of Naval Material (MAT 08Y) on April 27, 1981. Its purpose was to coordinate current and planned software engineering environment (SEE) initiatives in the Naval Systems Commands to ensure they are well integrated and non-redundant. Further, it was to provide a NAVMAT focal point for addressing software engineering environment issues, and to coordinate action on critical SEE related efforts.

The specific objectives defined in the SEEWG charter were:

- Provide a focal point for SEE activities within the Naval Material Command (NAVMAT).
- Coordinate current and planned SEE projects within NAVMAT to ensure consistency, compatibility, and non-redundancy of effort.
- Develop a specification for Navy SEEs that is based on the Navy Minimum Ada Programming Support Environment (MAPSE).
- Guide the assimilation and integration of standard SEEs into the Navy software development process.

Phase I of the SEEWG ended in March, 1982, with the publication of the "Report of the NAVMAT Software Engineering Environment Working Group." This report contains three parts: an executive summary; a framework for a SEE including a definition of a SEE and the description of an abbreviated software engineering process; and a method for a SEE's implementation and introduction into the Navy. This report meets the objectives of the SEEWG charter.

Plans for phase II of the SEEWG involve implementing the concepts discussed in the SEEWG report.

D.2.2.3 Description of Activity: The SEEWG members represent a number of the principal software engineering projects within the Navy. Their common effort to define the nature of a standard SEE for the Navy has received significant recognition in both the Navy software development world and in industry. Momentum towards a new software engineering process for the Navy has been created which can be exploited to assure the development and introduction of new tools and techniques. NAVMAT and a re-chartered SEEWG in Phase II will manage this process to capitalize on the progress thus far.

The objectives of developing requirements for a MAPSE-based SEE is addressed in the SEEWG report section entitled "Framework for a Navy Standard Software Engineering Environment." It should be noted that the Framework proposed is preliminary to formal specification of requirements for acquisition purposes.

The objectives of guiding assimilation and integration of standard SEE's into the Navy software development process has been addressed both in the Framework document and in the section entitled "Evolution Plan for a Navy Standard Software Engineering Environment." This SEEWG product deals specifically with the transition process from the current situation to the future standard SEE and also with subsequent technological innovation.

Thus, the two tangible products of the SEEWG effort are the Framework and Evolution Plan documents.

The main conclusions of each are summarized as follows.
Framework for a Navy Standard SEE:

- The SEE must support the entire software life cycle. A life cycle model has been developed in which to view software development as an incremental process.

- The SEE must be methodology-driven. A collection of tools is of little benefit in the absence of an integrating discipline for their development.

- A comprehensive set of tools and techniques to support each activity in the software life cycle have been identified.

- The information products derivable in each activity of the life cycle have been categorized according to whether the information must be baselined in the SEE data base or not. Baselined products are configuration managed and persist over the life of the software system being developed. Thus, the nature of the environment's data base is fundamental to the choice of tools provided and to the integrating discipline for their employment.

Evolution Plan for a Navy Standard SEE:

- The Navy Standard Software Engineering Environment (NSSEE) will be based on the Minimum Ada programming Support Environment (MAPSE).

- Existing Navy application projects and support systems cannot be ignored and should be able to gain some benefit from the concepts and products resulting from the definition and development of the NSSEE. An approach is described to assure the smooth transition to a modern Navy standard SEE.

- The Navy must build at least one integrated tool set that will operate through all phases of the life cycle model. The availability of such a Navy standard SEE and the control afforded by the baselining of standardized information products will greatly improve the Navy's ability to manage the acquisition and maintenance of effective software systems at reduced life cycle costs.

- Current policies, standards, and guidelines need to be changed in order to provide a framework within which a NSSEE can be built and used. Evolution of these policies, standards, and guidelines will be necessary as the Navy standard SEE evolves.

- Training/education is extremely important for managers, developers, and users of the NSSEE. Understanding how to use the Navy Standard SEE competently is crucial to deriving any benefits from it.

- The need for a research and development effort is identified to ensure the timely maturing of software technology to support full implementation of the desired Navy Standard SEE.

Perhaps the most important result of the SEEWG effort, however, may be the attention it has drawn to the need for a standard software engineering process to support the effective use of the Ada programming language in Navy applications. Momentum for a modern Navy standard SEE has begun. Action by NAVMAT will now be taken to see that the SEEWG recommendations give rise to actual specifications for a NSSEE and to policies and standards appropriate to support it.

D.2.2.4 Index to Problem Areas: A.1, A.2, A.5, B.1, B.2, B.3, B.4, C.3, C.4, D.1

D.2.3 Coordination of Program 6.2 Funds within NAVMAT

D.2.3.1 Responsible Activity: Naval Material Command and the three System Commands.

D.2.3.2 Scope of Initiative: The 6.2 program funds are used for advanced development of ideas, techniques, methodologies, etc. NAVMAT and four activities within NAVMAT are coordinating the planning and allocation of these funds for the maximum benefit to the Navy. These four activities are: the Office of Naval Research (ONR), the Naval Electronic Systems Command (NAVELEX), the Naval Air Systems Command (NAVAIR), and the Naval Sea Systems Command (NAVSEA).

D.2.3.3 Description of Activity: Program 6.1 funds are used for exploring the value of ideas applicable to tactical warfare programs (i.e., exploratory development). Program 6.2 funds are for further development of ideas which look most promising (i.e., advanced development). The ideas which then look most promising are implemented via 6.3 and 6.4 funding.

Program 6.2 funds within the Naval Material Command are being coordinated across four major NAVMAT activities. The objective of the 6.2 coordination effort is to determine the best use of advanced development (6.2) funds for NAVMAT. The focus is on today's and tomorrow's problems, either generic or specific. Many ideas proposed for 6.2 funding are evaluated for their long and short-term value, their cost versus their benefits, their technical feasibility, and their applicability to NAVMAT and its customers.

The 6.2 coordination effort is starting to achieve results. Users, who want to ensure their needs are heard, are becoming more and more involved in the Program 6 process. This ensures that funding is focused on known shortcomings in the systems and projected user's needs. Funding coordination, once focused only on 6.2, is starting to influence funding plans for 6.1 funds as well as 6.3 funds.

D.2.4 HQNAVMAT/SYSCOM Tactical Embedded Computer Software Impact Study (also referred to as the NAVMAT TECS Impact Study)

D.2.4.1 Responsible Activity: Chief of Naval Material

D.2.4.2 Scope of Initiative: The primary purpose of the HQNAVMAT/SYSCOM Tactical Embedded Computer Software Impact Study was to determine the impact of ECS software on NAVMAT field activities and to provide information required for decisions and planning to modify that impact. A secondary purpose was to provide a "snapshot" of the existing and planned future TEC software workload within NAVMAT. The study will cover ECS software and firmware, evaluate the criteria used to assign responsibilities for its development and support, and analyze the methods used to meet those responsibilities.

D.2.4.3 Description of Activity: The approach is to be incremental, with initial emphasis placed on major programs and their software. Other software will be examined generically, as required, to meet specific information needs.

Phase I

The 13 most involved field activities (see list below) will be evaluated. The survey will examine their present and anticipated future workload in ECS software. Survey information will be taken for currently supported systems and for those identified in Systems Command and field activity planning documents. Typical information sought includes:

Platform
System/subsystem
Major functions performed
Amount/size of embedded software
Amount/size of support software
Languages used
Type/cost of hardware support
Staff size and skill types
Contractor support
Facilities support and cost of operations

Phase II

After surveying the 13 field activities, the scope of the top three to five programs at each activity will be identified. Each of these high-impact software systems will be examined in greater detail. Typical additional information includes:

- History of involvement (how assigned)
- Related responsibilities
- Impact
- Cost drivers
- Relationship to activity roles and missions
- Relationship to technology base
- Recommendations and Suggestions

The 13 Field Activities are:

FCDSSA, DN	NCSC	NSRDC	NWC
FCDSSA, SD	NOSC	NTEC	PME-120-3 SSF*
NADC	NSWC	NUSC	PMTC
NATC			

*Included as a field activity for the purposes of this study

PLANNED ACTIONS AND MILESTONES

Phase I

		Date
1)	Letters to SYSCOMS requesting team member designation	14 May 82
2)	SYSCOM team members designated	28 May 82
3)	Team meeting	1 June 82
4)	Final data requirements established	18 June 82
5)	Survey and data base development completed	18 Aug 82
6)	Survey data analysis completed; initial report	1 Sept 82

Phase II

7)	Designation of high-impact systems	18 Aug 82
8)	Survey of high-impact systems completed	17 Sept 82
9)	Analysis of high-impact system data completed; initial report	1 Oct 82
10)	Review with selected SYSCOM sponsors completed	22 Oct 82
11)	Field activity visits completed	10 Dec 82
12)	Final data analysis completed, report completion	7 Jan 83
13)	Draft recommendations	1 Feb 83
14)	Briefings and final recommendations	1 Mar 83

D.2.5 Embedded Computer Systems Support Improvement Program (ESIP)

D.2.5.1 Responsible Activity: HQ AFLC/LOE

D.2.5.2 Scope of Initiative: This program addresses all categories of embedded computer systems in the Air Force (operational flight programs, electronic warfare, communications electronics, automatic test equipment and aircrew training devices). It is to focus funds and management on AFLC ECS projects that are not weapon system specific. The objective is to improve the support of embedded computer systems through technical innovations, applied management techniques, and more efficient information distribution. This effort has been estimated to require about 450 man/years spread over six years.

D.2.5.3 Description of Activity: AFLC's requirements for support of embedded computer systems have, for the most part, been addressed on a case-by-case basis. AFLC has provided a viable support environment for these systems as they arose. This system specific approach can be thought of as the first phase of AFLC ECS support. The numbers of ECS systems requiring support in the future and resource constraints preclude continuation of this approach, so the Defense and Space Systems Group of TRW, Inc. was asked to analyze ECS support requirements and suggest a more effective posture. This two year effort (FY79-80) has been completed and a number of specific recommendations have been made by TRW to HQ AFLC. The next step is to implement appropriate recommendations. The implementation will form the basis of what is to be known as the AFLC ECS Support Improvement Program (ESIP). The ESIP is the second phase of AFLC ECS support. The TRW study recommendations often overlap or enhance existing or planned projects. A dedicated effort must be made to coordinate the TRW initiatives with those of other programs. Examples of other programs are the AFLC ECS Statement of Operational Need (SON) for Embedded Computer Systems Software Support; PE 64740F, Computer Resources Technology; PE 6372B Advanced Computer Technology, and the OUSDR&E Software Technology Initiatives.

The Embedded Computer System (ECS) Support Improvement Program (ESIP) is an engineering and management methods development program. It addresses the various aspects of ECS support: responsiveness, readiness, quality, and cost effectiveness. The ESIP is the primary vehicle for applying funds and management to AFLC ECS projects that are not weapon system specific. The ESIP arises primarily from two documents: the AFLC Statement of Need (SON) for Embedded Computer System Software Support, and the Long Range Plan for Embedded Computer Systems Support by TRW dated Oct 81. The ESIP has been divided into five project areas.

- ECS Support Networks: From this project will be developed communication technology to provide both inter and intra AFLC networks. The networks will be used to support engineering and management, to increase productivity and mission responsiveness, and to reduce training and travel costs. The local and command-wide communication links will support: (1) The ECS change process. (2) Large data bases. (3) Automated standardized tools. (4) Intelligence handling. (5) Rapid reprogramming. (6) Software repositories. (7) Modular Integrated Support Facilities. (8) Training and education.

- Automation and Standardization of ECS Support Processes: From this project area will be developed automated and standardized tools to support the ECS change process. This project is necessary to improve the current practice of using manpower intensive and non-standard support tools, procedures, and nomenclatures. The automated standardized ECS support tools will be used for: (1) Management (2) Documentation (3) Analysis (4) Specification (5) Software development (6) Testing (7) Local development projects.

- Extendable Integration Support Facilities (EISF): From this project area will be developed the technology to expand and modularize the equipment capability of the AFLC Integration Support Facilities (ISFs). The project will establish standard ISF hardware modules, interfaces, and architectures. It will also expand and extend ISF capabilities to multiple systems and multiple functions. The modular EISF will be applied to support: (1) Combat mission readiness. (2) Weapon system growth and planned product improvements. (3) Multiple systems with dissimilar languages, and input/output requirements. (4) Multiple functions with common modules. (5) Automated standardized tools. (6) Training.

- Readiness Engineering: From this project area will be developed intelligence and data handling capabilities to enhance or establish support capabilities for preemptive engineering to respond to electronic threats. The project will provide for: (1) Defining ECS readiness related change requirements (2) An assessment of system and subsystem vulnerability. (3) Selection and implementation of changes. (4) Documentation and distribution of changes.

- Engineering Practices: From this project area will be developed a management approach to the support of ECS that will optimize technical skill and equipment. The project will be accomplished through administrative actions and specific contractor tasks. The project will investigate. (1) Developing an organization where engineers are matrixed into other management functions. (2) Developing ECS career progression paths. (3) Developing ECS training and professional education through a structured program which cycles hardware and software engineers through formal education courses and controlled job training programs. (4) Developing the use of standardized multi-use environmental simulations and system dynamics models in integration support facilities and aircrew training devices. (5) Developing supportability standards for

project will provide for: (1) Defining ECS readiness related change requirements. (2) An assessment of system and subsystem vulnerability. (3) Selection and implementation of changes. (4) Documentation and distribution of changes.

- Engineering Practices: From this project area will be developed a management approach to the support of ECS that will optimize technical skill and equipment. The project will be accomplished through administrative actions and specific contractor tasks. The project will investigate. (1) Developing an organization where engineers are matrixed into other management functions. (2) Developing ECS career progression paths. (3) Developing ECS training and professional education through a structured program which cycles hardware and software engineers through formal education courses and controlled job training programs. (4) Developing the use of standardized multi-use environmental simulations and system dynamics models in integration support facilities and aircrew training devices. (5) Developing supportability standards for implementing testability in the design phase and for acquiring documentation. (6) Developing capability for multi-ECS support through use of standard interfaces and local and command-wide ECS support software.

D.2.5.4. Index to Problems Areas: A.1, A.2, A.3, A.4, A.5, B.1, B.2, B.3, B.4, C.1, C.2, C.3, C.4, D.1, D.2, D.3.

D.2.6 AFSC Vanguard Planning System

D.2.6.1 Responsible Activity: HQ AFSC/XR is responsible for the entire Vanguard system. HQ AFSC/ALR is responsible for the Vanguard Functional Plan for Computer Resources

D.2.6.2 Scope of Initiative: Vanguard is an integrated research, development, and acquisition planning system developed by AFSC for their use.

"A major goal of Vanguard is to integrate technology base activities into the appropriate plans. Advanced development activities, and some exploratory development, can be directly tied to the mission area plans. Basic research and the remaining exploratory development programs are included in the technology base functional plan." (p. 2, AFSCP 80-3.)

D.2.6.3 Description of Activity:

a. The Vanguard planner identifies the jobs that must be done and assesses current Air Force capabilities to perform them. This assessment reveals where deficiencies exist. The descriptions of what must be done to satisfy these deficiencies are called development goals. The development goals are ranked based on a combination of the job deficiency and its importance. Listing the development goals in priority order provides the basis for planning development programs. This set of steps is called the analysis phase. The planner then assembles the baseline program, which consists of current and planned systems. Where there are continuing deficiencies, the planner proposes ways to correct them by recommending the development of new systems or the modification of existing systems. These recommendations include the development of the technology base required by these systems. The assemblage of approved and recommended programs is called the synthesis phase. The planner then prepares a briefing to present this information. This is the final form of the Vanguard plan.

b. There are three types of Vanguard plans. The first type deals with mission areas. Here the mission area planner aligns various systems against needs to determine deficiencies and proposes programs to satisfy

these deficiencies. The mission area viewpoint is not the total answer. It does not address questions such as commonality, compatibility, inter-operability, standardization, or austerity. Functional and force element plans, which cut across the mission areas, are also required. Force element plans deal with specific categories of forces, such as missiles and bombers. Functional area plans deal with the broad functions that are required of systems, such as propulsion, armament, avonics, etc. In contrast to mission area and force element plans, the topics of functional plans may change from time to time, depending upon the degree of management attention, the problems in the functional area, and the potential benefits from reviewing the functional area. The combination of the three types of plans gives an all encompassing view and perspective to the planning process.

c. Vanguard plans are used in the budget formulation process to give decision makers a complete perspective by answering such questions as -

- How does a program element (or a set of program elements) contribute to meeting Air Force military needs? The analytical procedure used in Vanguard helps to illuminate or make visible a program element's contribution to explicitly stated needs. Low payoff areas are identified which might be cut back to fund areas offering a higher return. Vanguard can help provide a basis for more rational budgeting.

- What is the contribution of a program to other mission areas? Many programs cut across mission areas. For budgetary purposes, only one mission area panel will rank a program. Vanguard planning enables panel members to realize contributions of a program to other mission areas.

- What will happen if we cancel or delay this program? This program may be required for the successful completion of other programs. By the same token, canceling this program might eliminate the need for other programs that feed it. Because of the way Vanguard information is structured and presented pictorially, consequences of program actions are readily apparent.

- How much will the program cost? Vanguard plans show both the estimated RDT&E and acquisition funding needed to complete the program.

- What are the key decision points and when do they occur? Vanguard graphically displays the major milestones in the life cycle of a system. These range from conception through initial operational capability (IOC) to phaseout.

- Have standardization, inter-operability, compatibility, and austerity been addressed? Functional and force element plans enable these questions to be addressed. For example, planners might need to know if a new avionics system will fit in or is compatible with the F-16. Using Vanguard plans, they could determine what other changes are being planned and check for compatibility.

d. The draft Vanguard Functional Plan for Computer Resources (1982) identifies nine development goals. Each goal is supported by a set of needs and assumptions. The nine goals are:

- User-system interface
- Software productivity
- Education and training
- Radiation hardening
- Signal processing
- Standardization
- High Order Language
- Security
- Distributed systems

D.2.4 Index to Problem Areas: A.5, B.2, B.4, C.1, C.3, C.4, D.1, D.2.

D.2.7 Post Deployment Software Support

D.2.7.1 Responsible Activity: DARCOM

D.2.7.2 Scope of Initiative: This program addresses the problem of providing effective and economic software support for the projected large number of Battlefield Automated Systems (BASs) which include embedded computers and which are to be deployed by the U.S. Army. Post Deployment Software Support (PDSS) is that part of the overall system support to sustain, modify and improve a deployed system's computer software as defined by the user or his representative. Computer software includes programs, instructions and data required to carry out computations or control functions. It includes the associated documentation. Computer software is divided into two general categories: fielded software and support software. Fielded software is the software that is deployed in and with the tactical equipment. Support software is the software that is deployed in and with the tactical equipment. Support software is the software used to develop and maintain the fielded software.

A study of the PDSS problem was initiated in July, 1978, when the U.S. Army Material Development and Readiness Command (DARCOM) tasked the U.S. Army Communictions Research and Development Command (CORADCOM) (now CECOM) to develop an Army-wide plan for PDSS with support from the other Army Commands.

A task force was created in August 1978 from representatives of Army staff agencies, Army Commands and Army project managers.

D.2.7.3 Description of Activity: Historically, PDSS planning by the system project manager has been performed on a system-by-system basis. This has resulted in a wide range of support strategies with varying degrees of responsiveness to the user. Although this approach may provide an optimal solution on a system basis, it provides a supoptimal solution on an Army-wide basis because of the potential resource savings possible through the economy-of-scale obtained by centralized support. In May 1978, DARCOM initiated action to provide a systematic approach for the planning of PDSS on an Army-wide basis. PDSS was also a topic of discusison at Battlefield Automation appraisals III and IV, 1978 and 1979, where the need for software and software standardization was cited.

A task force of representatives from Army staff agencies, Army commands, and Army project managers was formed to assist CORADCOM in defining the PDSS problem, identifying PDSS requirements, forming assumptions, and developing PDSS implementation alternatives, evaluation criteria, and a selection methodology.

The PDSS Concept Plan Report: The work of the task force resulted in the preparation of a PDSS concept Plan for BASs dated May 1980.

The PDSS Plan includes the setting up of eleven (11) centers for performing PDSS, and a recommended policy to improve the PDSS environment, both before and after deployment. The plan primarily addresses Post Deployment Software Support and U.S. Army Training and Doctrine Command (TRADOC) requirements for PDSS related functions. In addition to the plan, the report documents the findings of the PDSS task force study.

The report is structured in seven sections as follows:

Section 1. This section describes the purpose of the PDSS study effort, the applicability of the PDSS Concept Plan, a summary of the roles and missions for software support of BASs by DARCOM as the Materiel Developer (MD) and by TRADOC as the battlefield architect and principle Combat Developer (CD). It includes the background of the PDSS planning effort, assumptions for PDSS planning, and guidelines for development of the PDSS Concept Plan.

Section 2. Includes a discusison of the deficiencies in the system development process; problems due to proliferation of hardware, software, languages and PDSS organizations; other finding of the study effort, including problems in supporting the combat development process and problems after fielding.

Section 3. Discusses the structure of software to be supported, including the fielded subsystem and the support subsystem. It formulates a minimum set of tasks necessary for PDSS. It develops a software support organization and facilities model and discusses management considerations for implementation of PDSS.

Section 4. This section contains summary descriptions of eight different support concept alternatives considered including:

- Decentralize by system
- Centralize by equipment function
- Centralize by Battlefield Functional Area (BFA) at the Readiness Commands, or at the Doctrine Commands, or at the Developing Command.
- Adopt total centralization
- Follow a hybrid approach

Section 5. Introduces the recommended approach for PDSS based upon a hybrid approach; identification of 11 PDSS centers and responsible commands for those centers; identification of 91 Battlefield Automation systems (BASs) to be supported by the centers, categorized in one of three system categories, i.e., (1) large evolutionary systems, (2) small evolutionary and large stable systems, and (3) small stable systems. This section also identifies center management responsibilities; assignment of BASs among centers; requirements for PDSS and policy recommendations for effective PDSS.

Section 6. Is an estimate of resources needed for PDSS. This section is not included in the PDSS Concept Plan report and will be prepared separately.

Section 7. This section identifies 14 follow-on actions for implementation of the recommended concept plan and also identifies and briefly describes nine follow-on efforts required to complete the PDSS study.

D.2.7.4 Index to Problem Areas: A.1, A.4, A.5, B.2, B.4, D.1, D.2, D.3.